Discovering the Roots: Uniform Closure Results for Algebraic Classes under Factoring*

Pranjal Dutta
Chennai Mathematical Institute
Chennai, India
pranjal@cmi.ac.in

Nitin Saxena
IIT, Kanpur
Kanpur, India
nitin@cse.iitk.ac.in

Amit Sinhababu
IIT, Kanpur
Kanpur, India
amitks@cse.iitk.ac.in

ABSTRACT

Newton iteration (NI) is an almost 350 years old recursive formula that approximates a simple root of a polynomial quite rapidly. We generalize it to a matrix recurrence (allRootsNI) that approximates *all* the roots simultaneously. In this form, the process yields a better circuit complexity in the case when the number of roots r is small but the multiplicities are exponentially large. Our method sets up a linear system in r unknowns and iteratively builds the roots as formal power series. For an algebraic circuit $f(x_1, \ldots, x_n)$ of size s we prove that *each* factor has size at most a polynomial in: s and the degree of the squarefree part of f. Consequently, if f_1 is a $2^{\Omega(n)}$-hard polynomial then any nonzero multiple $\prod_i f_i^{e_i}$ is equally hard for *arbitrary* positive e_i's, assuming that $\sum_i \deg(f_i)$ is at most $2^{O(n)}$.

It is an old open question whether the class of poly(n)-sized formulas (resp. algebraic branching programs) is closed under factoring. We show that given a polynomial f of degree $n^{O(1)}$ and formula (resp. ABP) size $n^{O(\log n)}$ we can find a similar size formula (resp. ABP) factor in randomized poly($n^{\log n}$)-time. Consequently, if determinant requires $n^{\Omega(\log n)}$ size formula, then the same can be said about any of its nonzero multiples.

As part of our proofs, we identify a new property of multivariate polynomial factorization. We show that under a random linear transformation τ, $f(\tau\overline{x})$ *completely* factors via power series roots. Moreover, the factorization adapts well to circuit complexity analysis. This with allRootsNI are the techniques that help us make progress towards the old open problems; supplementing the large body of classical results and concepts in algebraic circuit factorization (eg. Zassenhaus, J.NT 1969; Kaltofen, STOC 1985-7 & Bürgisser, FOCS 2001).

CCS CONCEPTS

• **Theory of computation → Algebraic complexity theory**; *Problems, reductions and completeness*; • **Computing methodologies → Algebraic algorithms**; **Hybrid symbolic-numeric methods**; • **Mathematics of computing** → Combinatoric problems;

*Full version here: https://www.cse.iitk.ac.in/users/nitin/papers/factor-closure.pdf.

KEYWORDS

circuit factoring, formula, ABP, randomized, hard, VF, VBP, VP, VNP, quasipoly.

ACM Reference Format:
Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. 2018. Discovering the Roots: Uniform Closure Results for Algebraic Classes under Factoring. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3188745.3188760

1 INTRODUCTION

Algebraic circuits provide a way, alternate to Turing machines, to study computation. Here, the complexity classes contain (multivariate) polynomial families instead of languages. It is a natural question whether an algebraic complexity class is closed under factors. This is also a useful, and hence, a very well studied question both from the point of view of practice and theory. We study the following two questions related to multivariate polynomial factorization: **(1)** Let $\{f_n(x_1, \ldots, x_n)\}_n$ be a polynomial family in an algebraic complexity class C (egs. VP, VF, VBP, VNP or $\overline{\text{VP}}$ etc.). Let g_n be an arbitrary factor of f_n. Can we say that $\{g_n\}_n \in C$? Equivalently, is the class C *closed under factoring*? **(2)** Can we design an *efficient*, i.e. randomized poly(n)-time, algorithm to output the factor g_n with a representation in C? (*Uniformity*)

Different classes give rise to new challenges for the closure questions. Before discussing further, we give a brief overview of the algebraic complexity classes relevant for our paper. For more details, see [13, 52, 66].

Algebraic circuit is a natural model to represent a polynomial compactly. An *algebraic circuit* has the structure of a layered directed acyclic graph. It has leaf nodes labelled as input variables $x_1, \ldots, x_n$ and constants from the underlying field $\mathbb{F}$. All the other nodes are labelled as addition and multiplication gates. It has a root node that outputs the polynomial computed by the circuit. Some of the complexity parameters of a circuit are *size* (number of edges and nodes), *depth* (number of layers), syntactic *degree* (the maximum degree polynomial computed by any node), *fan-in* (maximum number of inputs to a node) and *fan-out*. An *algebraic formula* is a circuit whose underlying graph is a *directed tree*. In a formula, the fan-out of the nodes is at most one, i.e. 'reuse' of intermediate computation is not allowed.

The class VP (resp. VF) contains the families of n-variate polynomials of degree $n^{O(1)}$ over $\mathbb{F}$, computed by $n^{O(1)}$-sized circuits (resp. formulas). The class VF is sometimes denoted as VP$_e$, for it collects 'expressions' which is another name for formulas. Similarly, one can define VQP (resp. VQF) which contains the families of n-variate polynomials of degree $n^{O(1)}$ over $\mathbb{F}$, computed by

$2^{\text{poly}(\log n)}$-sized circuits (resp. formulas). If we relax the condition on the degree in the definition of VP, by allowing the degree to be possibly exponential, then we define the class VP_{nb}. Such circuits can compute constants of exponential bit-size (unlike VP).

Algebraic branching program (ABP) is another model for computing polynomials [66]. The class VBP contains the families of polynomials computed by $n^{O(1)}$-sized ABPs. We have the easy containments: VF $\subseteq$ VBP $\subseteq$ VP $\subseteq$ VQP = VQF [5, 73].

Finally, we give an overview of the class VNP, which can be seen as a non-deterministic analog of the class VP. A family of polynomials $\{f_n\}_n$ over $\mathbb{F}$ is in VNP if there exist polynomials $t(n), s(n)$ and a family $\{g_n\}_n$ in VP such that for every n, $f_n(\overline{x}) = \sum_{w \in \{0,1\}^{t(n)}} g_n(\overline{x}, w_1, \ldots, w_{t(n)})$. Here, *witness* size is $t(n)$ and *verifier* circuit g_n has size $s(n)$. VP is contained in VNP and it is believed that this containment is strict (Valiant's Hypothesis [71]).

Newton iteration based numerical methods are very popular in engineering [8, 25, 58]. This work introduces a new process to approximate all the roots of a circuit assuming that they are few and their multiplicites are known. This is based on a matrix recurrence, which in turn is derived from a new identity (Claim 6). Based on the process (called *allRootsNI* in Section 1.3) we get several consequences in very high-degree circuit factoring (eg. Theorem 1):

Every factor of a given circuit C has size polynomial in: size(C) and the degree of the squarefree part of C (note that it may be exponentially smaller than deg(C)).

and in factoring other poly-degree algebraic models (eg. Theorems 3, 14 & 15):

Each factor of a degree-d polynomial with VF (resp. VBP, VNP) complexity s, has VF (resp. VBP, VNP) complexity poly($s, d^{\log d}$). The latter is poly(s) if degree $d = 2^{O(\sqrt{\log s})}$.

Now, we briefly discuss the state of the art on the closure questions for various algebraic complexity classes. To cover more depth and breadth, see [21, 37, 38].

1.1 Previously Known Closure Results

Famously, Kaltofen [33–36] showed that VP is *uniformly* closed under factoring, i.e. for a given d degree n variate polynomial f of circuit size s, there exists a randomized poly(snd)-time algorithm that outputs its factor as a circuit whose size is bounded by poly(snd). This fundamental result has several applications such as 'hardness versus randomness' in algebraic complexity [2, 3, 19, 32], derandomization of Noether Normalization Lemma [53], in the problem of circuit reconstruction [40, 67], and polynomial equivalence testing [41]. In general, multivariate polynomial factoring has several applications including decoding of Reed-Solomon, Reed-Muller codes [29, 69], integer factoring [50], primary decomposition of polynomial ideals [24] and algebra isomorphism [30, 42].

It is natural to ask whether Kaltofen's VP factoring result can be extended to VP_{nb} which allows degree of the polynomials to be exponentially high. It is known that *not every* factor of a high degree polynomial has a small sized circuit. For example, the polynomial $x^{2^s} - 1$ can be computed in size s, but it has factors over $\mathbb{C}$ that require circuit size $\Omega\left(2^{s/2}/\sqrt{s}\right)$ [51, 64]. It is conjectured [12, Conj.8.3] that *low* degree factors of high degree small-sized circuits have *small* circuits. Partial results towards it are known. It was shown in [35] that if polynomial f given by a circuit of size s factors as $g^e h$, where g and h are coprime, then g can be computed by a circuit of size poly(e, deg(g), s). The question left open is to remove the dependency on e. In the special case where $f = g^e$, it was established that g has circuit size poly(deg(g), size(f)). On the other hand, several algorithmic problems are NP-hard, eg. computing the degree of the squarefree part, gcd, or lcm; even in the case of supersparse univariate polynomials [62].

Now, we discuss the closure results for classes more restrictive than VP (such as VF, VBP etc.). Unfortunately, Kaltofen's technique [36] for VF will give a superpolynomial-sized factor formula; as it heavily *reuses* intermediate computations while working with linear algebra and Euclid gcd. The same holds for the class VBP. In contrast, extending the idea of [19], Oliveira [57] showed that an n-variate polynomial with *bounded individual degree* and computed by a formula of size s, has factors of formula size poly(n, s). Furthermore, it was established that for a given n-variate individual-degree-r polynomial, computed by a circuit (resp. formula) of size s and depth Δ, there exists a poly(n^r, s)-time randomized algorithm that outputs any factor of f computed by a circuit (resp. formula) of depth $\Delta + 5$ and size poly(n^r, s). We are not aware of any work specifically on VBP factoring, except a special case in [39]—it dealt with the elimination of a *single* division gate from skew circuits—and another special case result in [31] that was weakened later owing to proof errors.

Going beyond VP we can ask about the closure of VNP. Bürgisser conjectured [12, Conj.2.1] that VNP is closed under factoring. Kaltofen's technique [36] for factoring VP circuits does not yield the closure of VNP. After our paper, Chou, Kumar and Solomon [14] have confirmed that VNP is indeed closed under factors.

Looking at the Border. Recently, *approximative* algebraic complexity classes like $\overline{\text{VP}}$ [27] have become objects of interest, especially in the context of the geometric complexity program [26, 54, 55], but also in the framework that yields the fastest matrix multiplication algorithms ([48] surveys the recent developments). Interestingly, [53, Thm.4.9] shows that the following three fundamental concepts are tightly related mainly due to circuit factoring results: **1)** efficient blackbox polynomial identity testing (PIT) for $\overline{\text{VP}}$, **2)** strong lower bounds against $\overline{\text{VP}}$, and **3)** efficiently computing an 'explicit system of parameters' for the invariant ring of an explicit variety with a given group action.

The related algorithmic questions, including factorization of $\overline{\text{VP}}$ circuits, have been recently put in PSPACE [22, 28]. This is the best uniform derandomization result known currently.

$\overline{\text{VP}}$ contains families of polynomials of degree poly(n) that can be approximated (infinitesimally closely) by poly(n)-sized circuits. Bürgisser [9–11] discusses approximative complexity of factors, proving that low degree factors of high degree circuits have small approximative complexity. In particular, $\overline{\text{VP}}$ is closed under factoring [9, Thm.4.1]. Like the standard versions, closure of $\overline{\text{VF}}$ resp. $\overline{\text{VBP}}$ is an open question. Recently, it has been shown that $\overline{\text{VF}}$ = width-2-$\overline{\text{VBP}}$ [7] while classically it is false [4]. The new methods that we present extend nicely to approximative classes because of their analytic nature (Theorem 14).

We conclude by stating a few reasons why closure results under factoring are interesting and non-trivial. First, there are classes that are *not* closed under factors. For example, the class of sparse polynomials; as a factor's sparsity may blowup super-polynomially [75]. Closure under factoring indicates the robustness of an algebraic complexity class, as, it proves that all nonzero multiples of a *hard* polynomial remain hard. For this reason, closure results are also important for proving lower bounds on the power of some algebraic proof systems [23].

Finally, factoring is the key reason why PIT, for VP, can be reduced to very special cases, and gets tightly related to circuit lower bound questions (like VP≠VNP?). See [32, Thm.4.1] for whitebox PIT connection and [2] for blackbox PIT. One of the central reasons is: Suppose a polynomial $f(\overline{y})$ is such that for a nonzero size-s circuit C, $C(f(\overline{y})) = 0$. Then, using factoring results for low degree C, one deduces that f also has circuit size poly(s). This gives us the connection: *If we picked a "hard" polynomial f then $f(\overline{y})$ would be a hitting-set generator (hsg) for C* [32, Thm.7.7]. Our work is strongly motivated by the open question of proving such a result for size-s circuits C that have high degree (i.e. $s^{\omega(1)}$). Our first factoring result (Theorem 1) implies such a 'hardness to hitting-set' connection for arbitrarily high degree circuits C assuming that: the squarefree part C_{sqfree} of C has low degree. In such a case we only have to find a hitting-set for C_{sqfree} which, as our result proves, has low algebraic circuit complexity.

1.2 Our Results

Before stating the results, we describe some of the assumptions and notations used throughout the paper. Set $[n]$ refers to $\{1, 2, \ldots, n\}$. Logarithms are wrt base 2. For a polynomial f, size(f) refers to the smallest size of circuits computing f; it is the *algebraic circuit complexity* of f.

Field. We denote the underlying field as $\mathbb{F}$ and assume that it is of characteristic 0 and algebraically closed. For eg. complex $\mathbb{C}$, algebraic numbers $\overline{\mathbb{Q}}$ or algebraic p-adics $\overline{\mathbb{Q}}_p$. All the results partially hold for other fields (such as $\mathbb{R}, \mathbb{Q}, \mathbb{Q}_p$ or finite fields of characteristic >degree of the input polynomial). For a brief discussion on this issue, see Section 5.

Ideal. We denote the variables $(x_1, \ldots, x_n)$ as $\overline{x}$. The *ideal* $I := \langle \overline{x} \rangle$ of the polynomial ring will be of special interest, and its power ideal I^d, whose generators are all degree d monomials in n variables. Often we will reduce the polynomial ring modulo I^d (inspired from *Taylor series of an analytic function around* $\overline{0}$ [70]).

Radical. For a polynomial $f = \prod_i f_i^{e_i}$, with f_i's coprime irreducible nonconstant polynomials and multiplicity $e_i > 0$, we define the squarefree part as the *radical rad*$(f) := \prod_i f_i$.

What can we say about these f_i's if f has a circuit of size s? Our main result gives a good circuit size bound when rad(f) has small degree. A more general formulation (with u_0) is:

Theorem 1. *If $f = u_0 u_1$ is a nonzero product in the polynomial ring $\mathbb{F}[\overline{x}]$, with size(f) + size(u_0) $\leq s$, then* every *factor of u_1 has a circuit of size poly(s + deg(rad(u_1)))*.

Note that Kaltofen's proof technique in the VP factoring paper [36] does not extend to the *exponential* degree regime (even when degree of rad(f) is small) because it requires solving equations with $\deg_{x_i}(f)$ many unknowns for some x_i, where $\deg_{x_i}(f)$ denotes *individual degree* of x_i in f, which can be very high. Also, basic operations like 'determining the coefficient of a univariate monomial' become #P-hard in the exponential-degree regime [72].

The proof technique in Kaltofen's single factor Hensel lifting paper [35, Thm.2] works only in the perfect-power case of $f = g^e$. It can be seen that rad(f) "almost" equals $f/\gcd(f, \partial_{x_i}(f))$, but the gcd itself can be of exponential-degree and so one cannot hope to use [35, Thm.4] to compute the gcd either. Univariate high-degree gcd computation is NP-hard [61, 62].

Interestingly, our result when combined with [35, Thm.3] implies that every factor g of f has a circuit of size polynomial in: size(f), deg(g) and min{deg(rad(f)), size(rad(f))}. We leave it as an open question whether the latter expression is polynomially related to size(f).

Theorem 1 shows an interesting way to create *hard* polynomials. In the theorem statement let the size concluded be $(s + \deg(\text{rad}(u_1)))^e$, for some constant e. Suppose, $f_1(x_1, \ldots, x_n)$ that is 2^{cn}-hard, then any nonzero $f := \prod_i f_i^{e_i}$ is also $2^{\Omega(n)}$-hard for *arbitrary* positive e_i's, as long as $\sum_i \deg(f_i) \leq 2^{\frac{cn}{e}-1}$.

In general, for a high degree circuit f, rad(f) can be of high degree (exponential in size of the circuit). Ideally, we would like to show that every degree d factor of f has poly(size(f), d)-size circuit. The next theorem reduces the above question to a special kind of modular division, where the denominator polynomial may *not* be invertible but the quotient is well-defined (eg. $x^2/x \mod x$). All that remains is to somehow eliminate this kind of *non-unit division* operator (which we leave as an open question). Consider 'random' elements $\alpha_i, \beta_i \in_r \mathbb{F}$ and the corresponding random linear map $\tau : x_i \mapsto \alpha_i y + x_i + \beta_i$, $i \in [n]$, where y is a new variable apart from $x_1, \ldots, x_n$.

Theorem 2. *If nonzero $f \in \mathbb{F}[\overline{x}]$ can be computed by a circuit of size s, then any degree d factor of $f(\tau\overline{x})$ is of the form A/B* mod *$\langle \overline{x} \rangle^{d+1}$ where polynomials A, B have circuits of size poly(sd).*

Note that in Theorem 2, B may be non-invertible in $\mathbb{F}[\overline{x}]/\langle \overline{x} \rangle^{d+1}$ and may have a high degree (eg. 2^s). So, we cannot use the famous trick of Strassen to do division elimination here [68].

We prove uniform closure results, under factoring, for the algebraic complexity classes defined below. Let $s : \mathbb{N} \longrightarrow \mathbb{N}$ be a function. Define the class VF(s) to contain families $\{f_n\}_n$ such that n-variate f_n can be computed by an algebraic formula of size poly($s(n)$) and has degree poly(n). Similarly, VBP(s) contains families $\{f_n\}_n$ such that f_n can be computed by an ABP of size poly($s(n)$) and has degree poly(n). Finally, VNP(s) denotes the class of families $\{f_n\}_n$ such that f_n has witness size poly($s(n)$), verifier circuit size poly($s(n)$), and has degree poly(n).

Theorem 3. *The classes VF($n^{\log n}$), VBP($n^{\log n}$), VNP($n^{\log n}$) are all closed under factoring.*

Moreover, there exists a randomized poly($n^{\log n}$)-time algorithm that: for a given $n^{O(\log n)}$ sized formula (resp. ABP) f of poly(n)-degree, outputs $n^{O(\log n)}$ sized formula (resp. ABP) of a nontrivial factor of f (if one exists).

Remark. The "time-complexity" in the algorithmic part makes sense only in certain cases. For example, when $\mathbb{F} \in \{\mathbb{Q}, \mathbb{Q}_p, \mathbb{F}_q\}$, or when one allows computation in the BSS-model [6]. In the former case our algorithm takes poly($n^{\log n}$) bit operations (assuming that the characteristic is zero or larger than the degree; see Theorem 15 in Section 5.2).

It is important to note that Theorem 3 does not follow by invoking Kaltofen circuit factoring [36] and VSBR transformation [73] from circuit to log-depth formula. Formally, if we are given a formula (resp. ABP) of size $n^{O(\log n)}$ and degree poly(n), then it has factors which can be computed by a circuit of size $n^{O(\log n)}$ and depth $O(\log n)$. If one converts the factor circuit to a formula (resp. ABP), one would get the size upper bound of the factor formula to be a much larger $(n^{O(\log n)})^{\log n} = n^{O(\log^2 n)}$. Moreover, Kaltofen's methods crucially rely on the circuit representation to do linear algebra, division with remainder, and Euclid gcd in an efficient way; a nice overview of the implementation level details to keep in mind is [45, Sec.3].

Our proofs extend to the approximative versions $C(n^{\log n})$ for $C \in \{\overline{\text{VF}}, \overline{\text{VBP}}, \overline{\text{VNP}}\}$ as well (Theorem 14).

As before, Theorem 3 has an interesting lower bound consequence: If f has VF (resp. VBP resp. VNP) complexity $n^{\omega(\log n)}$ then any nonzero fg has similar hardness (for $\deg(g) \le \text{poly}(n)$).

In fact, the method of Theorem 3 yields a formula factor of size $s^e d^{2\log d}$ for a given degree-d size-s formula (e is a constant). This means– If determinant $\det_n$ requires $n^{a \log n}$ size formula, for $a > 2$, then *any* nonzero degree-$O(n)$ multiple of $\det_n$ requires $n^{\Omega(\log n)}$ size formula.

Similarly, if we conjecture that a VP-complete polynomial f_n (say the homomorphism polynomial in [17, Thm.19]) has $n^{a \log n}$ ABP complexity, for $a > 4$, then *any* nonzero degree-$O(n)$ multiple of f_n has $n^{\Omega(\log n)}$ ABP complexity.

1.3 Proof Techniques

We begin by describing the new techniques that we have developed. Since they also give a new viewpoint on classic properties, they may be of independent interest. The techniques are *analytic* at heart ([46] has a good historical perspective). The way they appear in algebra is through the *formal power series* ring $\mathbb{F}[[x_1, \ldots, x_n]]$. The elements of this ring are multivariate formal power series, with degree as precision. So, an element f is written as $f = \sum_{i=0}^{\infty} f^{=i}$, where $f^{=i}$ is the *homogeneous part* of degree i of f. In algebra texts it is also called the *completion* of $\mathbb{F}[x_1, \ldots, x_n]$ wrt the ideal $\langle x_1, \ldots, x_n \rangle$ (see [43, Chap.13]). The *truncation* $f^{\le d}$, i.e. homogeneous parts up to degree d, can be obtained by reducing modulo the ideal $\langle \overline{x} \rangle^{d+1}$. Here d is seen as the *precision* parameter of the respective approximation of f. First, we introduce a factorization pattern of a polynomial f, over the power series ring, under a random linear transformation. Next, we discuss how this factorization helps us to bound the size of factors of the original polynomial.

Power Series Complete Split. We are interested in the *complete* factorization pattern of a polynomial $f(x_1, \ldots, x_n)$. We can view f as a univariate polynomial in one variable, say x_n, with coefficients coming from $\mathbb{F}[x_1, \ldots, x_{n-1}]$. It is easy to see the connection between linear factors and the roots: $x_n - g$ is a factor of f iff $f(x_1, \ldots, x_{n-1}, g(x_1, \ldots, x_{n-1})) = 0$.

Of course, one should not expect that a polynomial always has a factor which is linear in one variable. But, if one works with an algebraically closed field, then a univariate polynomial completely splits into linear factors (also see the *fundamental theorem of algebra* [15, §2.5.4]). So, if we go to the algebraic closure of $\mathbb{F}(x_1, \ldots, x_{n-1})$, any multivariate polynomial which is monic in x_n will split into factors all linear in x_n. A representation of the elements of $\overline{\mathbb{F}(x_1, \ldots, x_{n-1})}$ as a finite circuit is impossible (eg. $\sqrt{x_1}$). On the other hand, we show in this work that *all* the roots (wrt a new variable y) are actually elements from $\mathbb{F}[[x_1, \ldots, x_n]]$, after a *random* linear transformation on the variables, $\tau : \overline{x} \mapsto \overline{x} + \overline{\alpha} y + \overline{\beta}$, is applied (Theorem 4). Note– By a random choice $\alpha \in_r \mathbb{F}$ we will mean that choose randomly from a fixed finite set $S \subseteq \mathbb{F}$ of appropriate size (namely $> \deg(f)$). This will be in the spirit of [65].

Our proof of the existence of power series roots is *constructive*, as it also gives an algorithm to find approximation of the roots up to any precision, using formal power series version of the Newton iteration method (see [13, Thm.2.31]). We try to explain the above idea using the following example. Consider $f = (y^2 - x^3) \in \mathbb{F}[x, y]$. Does it have a factor of the form $y - g$ where $g \in \mathbb{F}[[x]]$? The answer is clearly 'no' as $x^{3/2}$ does not have any power series representation in $\mathbb{F}[[x]]$. But, what if we shift x randomly? For example, if we use the shift $y \mapsto y, x \mapsto x + 1$. Then, by Taylor series around 1, we see that $(x + 1)^{3/2}$ has a power series expansion, namely $1 + \frac{3}{2}x + \frac{3/2 \times 1/2}{2!}x^2 + \ldots$.

Formally, Theorem 4 shows that under a random $\tau : \overline{x} \mapsto \overline{x} + \overline{\alpha} y + \overline{\beta}$ where $\overline{\alpha}, \overline{\beta} \in_r \mathbb{F}^n$, polynomial f can be factored as $f(\tau \overline{x}) = \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i}$, where $g_i \in \mathbb{F}[[\overline{x}]]$ with the constant terms $g_i(\overline{0})$ being distinct, $d_0 := \deg(\text{rad}(f))$ and $\gamma_i > 0$.

Reducing Factoring to Computing Power Series Root Approximations. Using the split Theorem 4, we show that multivariate polynomial factoring reduces to power series root finding up to certain precision. Following the above notation f splits as $f(\tau \overline{x}) = \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i}$. For all $t \ge 0$, it is easy to see that $f(\tau \overline{x}) \equiv \prod_{i=1}^{d_0} (y - g_i^{\le t})^{\gamma_i} \mod I^{t+1}$, where $I := \langle x_1, \ldots, x_n \rangle$. Note that there is a one-one correspondence, induced by τ, between the polynomial factors of f and $f(\tau \overline{x})$ ($\because$ τ is invertible and f is y-free). We remark that the leading-coefficient of $f(\tau \overline{x})$ wrt y is a nonzero element in $\mathbb{F}$; so, we call it *monic* (Lemma 23). Next, we show case by case how to find a *polynomial* factor of $f(\tau \overline{x})$ from the approximate power series roots.

Case 1- Computing a Linear Factor of the Form $y - g(\overline{x})$. If the degree of the input polynomial is d, all the non-trivial factors have degree $\le (d - 1)$. So, if we compute the approximations of all the power series roots (wrt y) up to precision of degree $t = d - 1$, then we can recover all the factors of the form $y - g(x_1, \ldots, x_n)$. Technically, this is supported by the uniqueness of the power series factorization (Proposition 1).

Case 2- Computing a Monic Non-linear Factor. Assume that a factor g of total degree t is of the form $y^k + c_{k-1} y^{k-1} + \ldots + c_1 y + c_0$, where for all i, $c_i \in \mathbb{F}[\overline{x}]$. Now this factor g also splits into linear (in y) factors above $\mathbb{F}[[\overline{x}]]$ and obviously these linear factors are also linear factors of the original polynomial $f(\tau \overline{x})$. So we have to take the right combination of some k power series roots, with their approximations (up to the degree t wrt $\overline{x}$), and take the product mod I^{t+1}. Note that if we only want to give an existential proof of the size bound of the factors, we need not find the combination of the power series roots forming a factor algorithmically. Doing it through brute-force search takes exponential time ($\binom{d}{k}$ choices). Interestingly, using a classical (linear algebra) idea due to Kaltofen, it can be done in randomized polynomial time. We will spell out the ideas later, while discussing the algorithm part of Theorem 3.

Once we are convinced that looking at approximate (power series) roots is enough, we need to investigate methods to compute them. We will now sketch two methods. The first one approximates all the roots *simultaneously* up to precision δ. The next ones approximate the roots *one at a time*. In the latter, multiplicity of the root plays an important role.

Recursive Root Finding via Matrices (allRootsNI). We *simultaneously* find the approximations of all the power series roots g_i of $f(\tau\overline{x})$. At each recursive step we get a better precision wrt degree. We show that knowing approximations $g_i^{<\delta}$, of g_i up to degree $\delta-1$, is enough to (simultaneously for all i) calculate approximations of g_i up to degree δ. This new technique, of finding approximations of the power series roots, is at the core of Theorem 1.

First, let us introduce a nice identity. From now on we assume $f(\overline{x},y) = \prod_i (y-g_i)^{\gamma_i}$ (i.e. relabel $f(\tau\overline{x})$). By applying the derivative operator ∂_y, we get a classic identity (which we call *logarithmic derivative identity*): $(\partial_y f)/f = \sum_i \gamma_i/(y-g_i)$. Reduce the above identity modulo $I^{\delta+1}$ and define $\mu_i := g_i(\overline{0}) \equiv g_i \mod I$. This gives us (see Claim 6):

$$\frac{\partial_y f}{f} = \sum_{i=1}^{d_0} \frac{\gamma_i}{y-g_i} \equiv \sum_{i=1}^{d_0} \frac{\gamma_i}{y-g_i^{<\delta}} + \sum_{i=1}^{d_0} \frac{\gamma_i \cdot g_i^{=\delta}}{(y-\mu_i)^2} \mod I^{\delta+1}.$$

In terms of the d_0 unknowns $g_i^{=\delta}$, the above is a linear equation. (Note- We treat γ_i,μ_i's as known.) As y is a free variable above, we can fix it to d_0 "random" elements c_i in $\mathbb{F}$, $i\in[d_0]$. One would expect these fixings to give a linear system with a unique solution for the unknowns. We can express the system of linear equations succinctly in the following matrix representation: $M \cdot v_\delta = W_\delta \mod I^{\delta+1}$. Here M is a $d_0 \times d_0$ matrix; each entry is denoted by $M(i,j) := \frac{\gamma_i}{(c_i-\mu_j)^2}$. Vector v_δ resp. W_δ is a $d_0 \times 1$ matrix where each entry is denoted by $v_\delta(i) := g_i^{=\delta}$ resp. $W_\delta(i) := \frac{\partial_y f}{f}\Big|_{y=c_i} - G_{i,\delta}$, where $G_{i,\delta} := \sum_{k=1}^{d_0} \gamma_k/(c_i - g_k^{<\delta})$. We ensure that $\{c_i,\mu_i \mid i\in[d_0]\}$ are distinct, and show that the determinant of M is non-zero (Lemma 24). So, by knowing approximations up to $\delta-1$, we can recover δ-th part by solving the above system as $v_\delta = M^{-1}W_\delta \mod I^{\delta+1}$. An important point is that the random c_i's will ensure: all the reciprocals involved in the calculation above do exist mod $I^{\delta+1}$.

Self-correction property: Does the above recursive step need an exact $g_i^{<\delta}$? We show the self correcting behavior of this process of root finding, i.e. in this iterative process there is no need to filter out the "garbage" terms of degree $\geq \delta$ in each step. If one has recovered g_i correct up to degree $\delta-1$, i.e. say we have calculated $g'_{i,\delta-1} \in \mathbb{F}(\overline{x})$ such that $g'_{i,\delta-1} \equiv g_i^{<\delta} \mod I^\delta$, and say we solve $M\tilde{v}_\delta = \widetilde{W}_\delta$ exactly, where $\widetilde{W}_\delta(i) := \frac{\partial_y f}{f}\Big|_{y=c_i} - \widetilde{G}_{i,\delta}$, and $\widetilde{G}_{i,\delta} := \sum_{k=1}^{d_0} \gamma_k/(c_i - g'_{k,\delta-1})$. Still, we can show that $g'_{i,\delta} := g'_{i,\delta-1} + \tilde{v}_\delta(i) \equiv g_i^{\leq\delta} \mod I^{\delta+1}$ (Claim 7). So, we made progress in terms of the precision (wrt degree).

Rapid Newton Iteration with Multiplicity. We show that from allRootsNI, we can derive a formula that finds $g_1^{<2^{t+1}}$ using *only* $g_1^{<2^t}$, i.e. the process has quadratic convergence and it does not involve roots other than g_1. Rewrite $\partial_y f/f = \sum_{i=1}^{d_0} \gamma_i/(y-g_i) = (1+L_1)\gamma_1/(y-g_1)$, where $L_1 := \sum_{1<i\leq d_0} \frac{\gamma_i}{y-g_i} \cdot \frac{y-g_1}{\gamma_1}$. This implies $f/\partial_y f = (1+L_1)^{-1} \cdot (y-g_1)/\gamma_1$. Now, if we put $y = y_t := g_1^{<2^t}$, then $y_t - g_i = g_1^{<2^t} - g_i$ is a unit in $\mathbb{F}[[\overline{x}]]$ for $i \neq 1$ ($\because$ it is a nonzero constant mod I). Also, $y_t - g_1 = g_1^{<2^t} - g_1 \equiv 0 \mod I^{2^t}$, implying $L_1|_{y=y_t} \equiv 0 \mod I^{2^t}$. Thus, $(L_1 \cdot (y-g_1))\Big|_{y=y_t} \equiv 0 \mod I^{2^{t+1}}$.

Hence, $f/\partial_y f\Big|_{y=y_t} = (y_t - g_1)/\gamma_1 \mod I^{2^{t+1}}$.

This shows that, if $f(\overline{x},y) = (y-g)^e h$, where $h|_{y=g} \neq 0 \mod I$ and $e>0$, then the power series for g can be approximated by the recurrence:

$$y_{t+1} := y_t - e \cdot \frac{f}{\partial_y f}\Bigg|_{y=y_t} \tag{1}$$

where $y_t \equiv g \mod I^{2^t}$. This we call a *generalized Newton Iteration* formula, as it works with any multiplicity $e>0$.

In fact, when $e=1$, g is called a *simple root* of f; the above is an alternate proof of the classical Newton Iteration (NI) [56] that finds a simple root in a recursive way (see Lemma 22). When *all* the roots are simple there are numerical methods to simultaneously approximate them [1, 18, 20, 44]. However, it is well known that NI *fails* to approximate the roots that repeat (see [49]). In that case either NI is used on the function $f/\partial_y f$ or, though less frequently, the generalized NI is used in numerical methods (see [16, Eqn.6.3.13]).

There is a technical point about our Eqn.1 when $e \geq 2$. The denominator $\partial_y f|_{y=y_t}$ is zero mod I, thus, its reciprocal does not exist! However, the ratio $(f/\partial_y f)\Big|_{y=y_t}$ does exist in $\mathbb{F}[[\overline{x}]]$. On the other hand, if $e=1$ then the denominator $\partial_y f|_{y=y_t}$ is nonzero mod I, thus, it is invertible in $\mathbb{F}[[\overline{x}]]$ and that is necessary for fast algebraic circuit computation (esp. division elimination).

We can compare the NI formula with the recurrence formula (which we call *slow* Newton Iteration) used in [19, Eqn.5], [57, Lem.4.1] for root finding. The slow NI formula is $Y_{t+1} = Y_t - \frac{f(\overline{x},Y_t)}{\partial_y f(\overline{0},Y_1)}$, where $Y_t \equiv g \mod I^t$. The rate of convergence of this iteration is linear, as it takes δ many steps (instead of $\log\delta$) to get precision up to degree δ. One can also compare NI with other widespread processes like multifactor Hensel lifting [74, Sec.15.5], [77] and the implicit function theorem paradigm [46, Sec.1.3], [47, 59]; however, we would not like to digress too much here as the latter concept covers a whole lot of ground in mathematics.

1.4 Proof Overview

In all our proofs, we use the reduction of factoring to power series root approximation, and then find the latter using various techniques described before.

Proof Idea of Theorem 1. We use the technique of allRootsNI to find the approximations of all the power series roots of $f(\tau\overline{x})$. As we already discussed how to find a polynomial factor g of u_1 (that divides f) from the roots of $f(\tau\overline{x})$, what remains is to analyze the size bound for power series roots that we get from allRootsNI process. We note a few crucial points that help to prove the size bound.

Let d_0 be the degree of $\mathrm{rad}(u_1)$. The number of distinct power series roots, of $u_1(\tau\overline{x})$ wrt y, is d_0. It suffices to approximate the power series roots up to degree d_0, as any nontrivial polynomial factor of $\mathrm{rad}(u_1(\tau\overline{x}))$ has degree less than d_0. Also, a size bound on these factors of the radical directly gives a size bound on the polynomial factor g.

The logarithmic derivative satisfies:

$$\partial_y \log f(\tau\overline{x}) = \partial_y \log u_0(\tau\overline{x}) + \partial_y \log u_1(\tau\overline{x}).$$

Since we have size s circuits for both f and u_0, and y is later fixed to random c_i's in $\mathbb{F}$, we can approximate the first two logarithmic derivative circuits modulo I^{d_0+1}. This approximates $\partial_y u_1(\tau\overline{x})/u_1(\tau\overline{x})$.

On this, allRootsNI process is used to approximate the power series roots of $u_1(\tau\overline{x})$ up to degree d_0. The self correcting behavior of the allRootsNI is crucial in the size analysis. If one had to truncate modulo I^{d_0+1} at each recursive step, there would have been a multiplicative blowup (by d_0) in each step, which would end up with an exponential blow up in the size of the roots. The self correcting property allows to complete allRootsNI process, with division gates and partially correct roots $g'_{i,\delta}$, to get a circuit of size $\mathrm{poly}(sd_0)$. The truncation modulo I^{d_0+1}, to get a root of degree $\leq d_0$, is performed only once in the end. See Section 4.1.

The steps in the proof of Theorem 1 are constructive. However, to claim that we have an efficient algorithm we will need, in advance, the multiplicity of each of the d_0 roots. It is not clear how to find them efficiently, even in the univariate case $n = 1$, as the multiplicity could be exponentially large.

Proof Idea of Theorem 2. The main technique used is NI with multiplicity. The main barrier in resolving high degree case is handling roots with high multiplicities (i.e. super-polynomial in size s). If all the roots of the polynomial have multiplicity equal to one, then we can use classical Newton iteration. If the multiplicity of a root is low (up to $\mathrm{poly}(s)$), we can differentiate and bring down the multiplicity to one. In Theorem 1, we handled the case of high multiplicity by assuming that the radical has small degree.

So, the only remaining case is when both the number of roots, and their multiplicities, are high. Newton iteration with multiplicity helps here. Note that we need to know the multiplicity of the root *exactly* to apply NI with multiplicity; here, we will simply guess them non-uniformly. In the end, the process gives a circuit of size $\mathrm{poly}(sd)$ with division gates, giving the root mod I^{d+1}. By using a standard method the division gates can all be pushed "out" to the root. See Section 4.2.

Proof Idea of Theorem 3. Here, we show that the algebraic complexity classes $VF(n^{\log n}), VBP(n^{\log n}), VNP(n^{\log n})$ are closed under factoring. In fact, we also give randomized $n^{O(\log n)}$-time algorithm to output the factors as formula (resp. algebraic branching program). The key technique here is the classical Newton Iteration. The crucial advantage of NI over other approaches of power series root finding is that NI requires only $\log d$ steps to get precision up to degree d, whereas allRootsNI, [19, Eqn.5] or [57, Lem.4.1] require d steps. This leads to a slower size blow up in the case of restricted models like formula or ABP.

In a formula resp. ABP, we cannot reuse intermediate computations. So each recursive step of NI incurs a blow up by d^2, as one needs to substitute y_t in a degree d polynomial $f(y)$ which may require that many copies of y_t-powers. But, as the NI process has only $\log d$ steps, ultimately, we get $d^{2\log d}$ blow up in the size bound. This is the main idea of the existential results in Theorem 3. Moreover, an interesting by-product is that VF, VBP and VNP are closed under factors if we only consider polynomials with individual degree *constant* (also see [57]).

All the steps in the proof of the existential result are algorithmically efficient except for one. We are recovering all the power series roots and multiplying a few of them to get a non-trivial factor. How do we choose the right combination of the roots which gives a non-trivial factor? If we search for the right combination in a brute-force way, it would need exponential (like 2^d) time complexity. Here, linear algebra saves us; the idea dates back to Kaltofen's algorithm for bivariate factoring. Our contribution lies in the careful analysis of the different steps, coming up with a new algorithm for computing gcd, and making sure that everything works with formulas resp. ABPs.

Consider the transformed polynomial $f(\tau\overline{x})$ that is monic and degree d in y. It will help us if we think of this polynomial as a bivariate (i.e. in y and a new degree-counter T). This somewhat reduces the problem to a two-dimensional case and makes the modular computations feasible (see [45, Sec.1.2.2]). So, we need to apply the map $\overline{x} \mapsto T\overline{x}$, where T is a new formal variable; call the resulting polynomial $\tilde{f}(\overline{x}, T, y)$. This map preserves the power series roots; in fact, we can get the roots of $f(\tau\overline{x})$ by putting $T = 1$. Now comes the most important idea in the algorithm. Approximate a root g_i up to large enough precision (say $k := 2d^2$). Solve the system of linear equations $u = (y - g_i^{\leq k}(Tx)) \cdot v \bmod T^{k+1}$ for monic polynomials u, v. Then, u will give a non-trivial factor when we compute $\gcd_y(u, \tilde{f})$. Intuitively, the gcd gives us the irreducible polynomial factor whose root is the power series g_i that we had earlier computed by NI.

Note that a modified gcd computation is needed to actually get a factor as a formula resp. ABP. If one uses the classical Euclidean algorithm, there are d recursive steps to execute; at each step there would be a blow up of d (as for formula or ABP, we cannot reuse any intermediate computation). So, in this approach (eg. the one used in [45]), gcd of the two formulas will be of exponential size. The way we achieve a better bound is by first using NI to approximate *all* the power series roots of u and $\tilde{f}$. Subsequently, we filter the ones that appear in both to learn the gcd. There is an alternate way as well based on our Claim 11. See Section 4.3.

2 PRELIMINARIES

In our proofs we will need some basic results about formulas, ABPs and circuits. In particular, we can efficiently eliminate a division gate, we can extract a homogeneous part, and we can compute a (first-order) derivative. Also, see [45, Sec.2].

Determinant is in VBP and is computable by a $n^{O(\log n)}$ size formula.

We will use properties of $\gcd(f, g)$ and a related determinant polynomial called *resultant*.

To save space we have moved the well known details to Sec. A.

3 POWER SERIES FACTORIZATION OF POLYNOMIALS

Instead of looking into the factorization over $\mathbb{F}[\overline{x}]$, we look into the factorization pattern of a polynomial over $\mathbb{F}[[x_1, \ldots, x_n]]$, namely, formal power series of n-variables over field $\mathbb{F}$. To talk about factorization, we need the notion of *uniqueness* which the following proposition ensures.

Proposition 1. *[76, Chap.VII] Power series ring $\mathbb{F}[[x_1, \dots, x_n]]$ is a unique factorization domain (UFD), and so is $\mathbb{F}[[\overline{x}]][y]$.*

As discussed before, we need to first apply a random linear map, that will make sure that the resulting polynomial splits completely over the ring $\mathbb{F}[[\overline{x}]]$. (Recall: $\mathbb{F}$ is algebraically closed.)

Theorem 4 (Power Series Complete Split). *Let $f \in \mathbb{F}[\overline{x}]$ with $deg(rad(f)) =: d_0 > 0$. Consider $\alpha_i, \beta_i \in_r \mathbb{F}$ and the map $\tau : x_i \mapsto \alpha_i y + x_i + \beta_i$, $i \in [n]$, where y is a new variable.*

Then, over $\mathbb{F}[[\overline{x}]]$, $f(\tau\overline{x}) = k \cdot \prod_{i\in[d_0]} (y - g_i)^{\gamma_i}$, where $k \in \mathbb{F}^*$, $\gamma_i > 0$, and $g_i(\overline{0}) := \mu_i$. Moreover, μ_i's are distinct nonzero field elements.

Proof. Let the irreducible factorization of f be $\prod_{i\in[m]} f_i^{e_i}$. We apply a random τ so that f, thus all its factors, become monic in y (Lemma 23). The monic factors $\tilde{f}_i := f_i(\tau\overline{x})$ remain irreducible ($\because$ τ is invertible). Also, $\tilde{f}_i(\overline{0}, y) = f_i(\overline{\alpha}y + \overline{\beta})$ and $\partial_y \tilde{f}_i(\overline{0}, y)$ remain coprime ($\because$ $\overline{\beta}$ is random). In other words, $\tilde{f}_i(\overline{0}, y)$ is square free.

In particular, one can write $\tilde{f}_1(\overline{0}, y)$ as $\prod_{i=1}^{\deg(f_1)} (y - \mu_{1,i})$ for distinct nonzero field elements $\mu_{1,i}$ (ignoring the constant which is the coefficient of the highest degree of y in $\tilde{f}_1$). Using classical Newton Iteration (see Lemma 22 or [13, Thm.2.31]), one can write $\tilde{f}_1(\overline{x}, y)$ as a product of power series $\prod_{i=1}^{\deg(f_1)} (y - g_{1,i})$, with $g_{1,i}(\overline{0}) := \mu_{1,i}$. Thus, each $f_i(\tau\overline{x})$ can be factored into linear factors in $\mathbb{F}[[\overline{x}]][y]$.

As f_i's are irreducible coprime polynomials, one can deduce that $\tilde{f}_i(\overline{0}, y)$, $i \in [m]$, are mutually coprime. In other words, $\mu_{j,i}$ are distinct and they are $\sum_i \deg(f_i) = d_0$ many. Hence, $f(\tau\overline{x})$ can be completely factored as $\prod_{i\in[m]} f_i(\tau\overline{x})^{e_i} = \prod_{i\in[d_0]} (y - g_i)^{\gamma_i}$, with $\gamma_i > 0$ and the field constants $g_i(\overline{0})$ being distinct. □

Corollary 5. *Suppose g is a polynomial factor of f. As before let $f(\tau\overline{x}) = \prod_{i\in[m]} f_i(\tau\overline{x})^{e_i} = k \cdot \prod_{i\in[d_0]} (y - g_i)^{\gamma_i}$. As $g(\tau x) \mid f(\tau\overline{x})$ we deduce that $g(\tau x) = k' \prod (y - g_i)^{c_i}$ with $0 \le c_i \le \gamma_i$.*

Moreover, we can get back g by applying τ^{-1} on the resulting polynomial $g(\tau\overline{x})$.

4 MAIN RESULTS

This section proves Theorems 1–3. The proofs are self contained and we assume for the sake of simplicity that the underlying field $\mathbb{F}$ is algebraically closed and has characteristic 0. When this is not the case, we discuss the corresponding theorems in Section 5.

4.1 Factors of a Circuit with Low-degree Radical: Proof of Theorem 1

In this section, we use Theorem 4 and allRootsNI to partially solve the case of circuits with exponential degree (stated in [34] and studied in [11, 35]).

Proof of Theorem 1. From the hypothesis $f = u_0 u_1$. Define $\deg(f) =: d$. Suppose $u_1 = h_1^{e_1} \dots h_m^{e_m}$, where h_i's are coprime irreducible polynomials. Let d_0 be the degree of $\mathrm{rad}(u_1) = \prod_i h_i$. Note that $\deg(h_i), m \le d_0$ and the multiplicity $e_i \le d \le s^{O(s)}$, where s is the size bound of the input circuit. Thus, to get the size bound of any factor of u_1, it is enough to show that for each i, h_i has a circuit of size $\mathrm{poly}(sd_0)$.

Using Theorem 4, we have $\tilde{f}(\overline{x}, y) := f(\tau\overline{x}) = k \cdot u_0(\tau\overline{x}) \cdot \prod_{i\in[d_0]} (y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) := \mu_i$ being distinct. From Corollary 5 we deduce that $h_i(\tau\overline{x}) = k_i \prod_{i\in[d_0]} (y - g_i^{\le d_0})^{\delta_i} \bmod I^{d_0+1}$, with ideal $I := \langle x_1, \dots, x_n \rangle$, exponent $\delta_i \in \{0, 1\}$ and nonzero $k_i \in \mathbb{F}$. We can get h_i by applying τ^{-1}. Hence, it is enough to bound the size of $g_i^{\le d_0}$.

Let $\tilde{u_0} := u_0(\tau\overline{x})$. From the repeated applications of Leibniz rule of the derivative ∂_y, we deduce, $\partial_y \tilde{f}/\tilde{f} = \partial_y \tilde{u_0}/\tilde{u_0} + \sum_{i=1}^{d_0} \gamma_i/(y - g_i)$. (Recall: $\partial_y(FG) = (\partial_y F)G + F(\partial_y G)$.)

At this point we move to the formal power series, so that the reciprocals can be approximated as polynomials. Note that $y - g_i$ is invertible in $\mathbb{F}[[\overline{x}]]$ when y is assigned any value $c_i \in \mathbb{F}$ which is not equal to μ_i. We intend to find $g_i \bmod I^\delta$ inductively, for all $\delta \ge 1$. We assume that μ_i's and γ_i's are known. Suppose, we have recovered up to $g_i \bmod I^\delta$ and we want to recover $g_i \bmod I^{\delta+1}$. The relevant recurrence, for $\delta \ge 1$, is:

Claim 6 (Recurrence). $\sum_{i=1}^{d_0} \gamma_i \cdot g_i^{=\delta}/(y - \mu_i)^2 \equiv \partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0} - \sum_i \gamma_i/(y - g_i^{<\delta}) \bmod I^{\delta+1}$.

Proof of Claim 6. Using a power series calculation (Lemma 25), we have $\frac{1}{y-g_i} \equiv \frac{1}{y-(g_i^{<\delta}+g_i^{=\delta})} \equiv \frac{1}{y-g_i^{<\delta}} + \frac{g_i^{=\delta}}{(y-\mu_i)^2} \bmod I^{\delta+1}$. Multiplying by γ_i and summing over $i \in [d_0]$, the claim follows. □

By knowing approximation up to the $\delta - 1$ homogeneous parts of g_i, we want to find the δ-th part by solving a linear system. For concreteness, assume that we have a rational function $g'_{i,\delta-1} := C_{i,\delta-1}/D_{i,\delta-1}$ such that $g'_{i,\delta-1} \equiv g_i^{<\delta} \bmod I^\delta$. Next, we show how to compute $g_i^{\le\delta}$.

We recall the process as outlined in allRootsNI (Section 1.3). In the free variable y, we plug-in d_0 random field value c_i's and get the following system of linear equations: $M \cdot v_\delta = W_\delta$, where M is a $d_0 \times d_0$ matrix with (i,j)-th entry, $M(i,j) := \gamma_j/(c_i - \mu_j)^2$. Column v_δ resp. W_δ is a $d_0 \times 1$ matrix whose i-th entry is denoted $v_\delta(i)$ resp. $(\partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0})|_{y=c_i} - \tilde{G}_{i,\delta}$, where $\tilde{G}_{i,\delta} := \sum_{j=1}^{d_0} \gamma_j/(c_i - g'_{j,\delta-1})$. Think of the solution v_δ as being both in $\mathbb{F}(\overline{x})^{d_0}$ and in $\mathbb{F}[[\overline{x}]]^{d_0}$; both the views help.

Now we will prove two interesting facts. First, M is invertible (Lemma 24). Second, define $g'_{i,0} := \mu_i$ and, for $\delta \ge 1$, $g'_{i,\delta} := g'_{i,\delta-1} + v_\delta(i)$. Then, $g'_{i,\delta}$ approximates g_i well:

Claim 7 (Self-correction). *Let $i \in [d_0]$ and $\delta \ge 0$. Then, $g'_{i,\delta} \equiv g_i^{\le\delta} \bmod I^{\delta+1}$.*

Proof of Claim 7. We prove this by induction on δ. It is true for $\delta = 0$ by definition. Suppose it is true for $\delta - 1$. This means we have $g'_{i,\delta-1} \equiv g_i^{<\delta} \bmod I^\delta$ for all i. Let us write $g'_{i,\delta-1} =: g_i^{<\delta} + A_{i,\delta} + A'_{i,\delta}$, where $A'_{i,\delta} \equiv 0 \bmod I^{\delta+1}$ and $A_{i,\delta}$ is homogeneous of degree δ. Hence, for $i \in [d_0]$, the linear constraint is: $\sum_{j=1}^{d_0} \gamma_j \cdot v_\delta(j)/(c_i - \mu_j)^2 \equiv \partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0} - \sum_j \gamma_j/(c_i - g'_{j,\delta-1}) \bmod I^{\delta+1}$.

The "garbage" term $A_{j,\delta}$ in RHS can be isolated using Lemma 25 as: $1/(c_i - g'_{j,\delta-1}) \equiv \frac{1}{c_i-(g_j^{<\delta}+A_{j,\delta})} \equiv 1/(c_i - g_j^{<\delta}) + A_{j,\delta}/(c_i - \mu_j)^2 \bmod I^{\delta+1}$. So, under modulo $I^{\delta+1}$, we get:

$$\sum_{j=1}^{d_0} \frac{\gamma_j \cdot v_\delta(j)}{(c_i-\mu_j)^2} \equiv \frac{\partial_y \tilde{f}}{\tilde{f}} - \frac{\partial_y \tilde{u_0}}{\tilde{u_0}} - \sum_{j=1}^{d_0}\left(\frac{\gamma_j}{c_i - g_j^{<\delta}} - \frac{\gamma_j \cdot A_{j,\delta}}{(c_i-\mu_j)^2}\right)$$

Rewriting this, using Claim 6, we get:

$$\sum_{j=1}^{d_0} \frac{\gamma_j}{(c_i-\mu_j)^2}\left(v_\delta(j) + A_{j,\delta}\right) \equiv \sum_{j=1}^{d_0} \frac{\gamma_j}{(c_i-\mu_j)^2} \cdot g_j^{=\delta} \bmod I^{\delta+1}.$$

Thus, $\sum_{j=1}^{d_0} \gamma_j \cdot (v_\delta(j) + A_{j,\delta} - g_j^{=\delta})/(c_i - \mu_j)^2 \equiv 0 \bmod I^{\delta+1}$. As we vary $i \in [d_0]$ we deduce, by Lemma 24, that $v_\delta(j) + A_{j,\delta} - g_j^{=\delta} \equiv 0 \bmod I^{\delta+1}$. Hence, $g'_{j,\delta} = g'_{j,\delta-1} + v_\delta(j) \equiv (g_j^{<\delta} + A_{j,\delta}) + (g_j^{=\delta} - A_{j,\delta}) = g_j^{\leq\delta} \bmod I^{\delta+1}$. This proves it for all $j \in [d_0]$. □

Size Analysis: Here we give the overall process of finding factors using allRootsNI technique and analyze the circuit size needed at each step to establish the size bound of the factors. As discussed before, we need to analyze only the power series root approximation $g_i^{\leq\delta}$ or $g'_{i,\delta}$.

At the $(\delta-1)$-th step of allRootsNI process, we have a multi-output circuit (with division gates) computing $g'_{i,\delta-1}$ as a rational function, for all $i \in [d_0]$. Specifically, let us assume that $g'_{i,\delta-1} =: C_{i,\delta-1}/D_{i,\delta-1}$, where $D_{i,\delta-1}$ is invertible in $\mathbb{F}[[\overline{x}]]$. So, the circuit computing $g'_{i,\delta-1}$ has a division gate at the top that outputs $C_{i,\delta-1}/D_{i,\delta-1}$. We would eliminate this division gate only in the end (see the standard Lemma 20). Now we show how to construct the circuit for $g'_{i,\delta}$, given the circuits for $g'_{i,\delta-1}$.

From $v_\delta = M^{-1}W_\delta$, it is clear that $\exists \beta_{ij}$ such that $v_\delta(i) = \sum_{j=1}^{d_0} \beta_{ij} W_\delta(j) = \sum_{j=1}^{d_0} \beta_{ij}\left((\partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0})|_{y=c_j} - \tilde{G}_{j,\delta}\right)$.

Initially we precompute, for all $j \in [d_0]$, $(\partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0})|_{y=c_j}$: Note that $\partial_y \tilde{f}$ has poly(s) size circuit (high degree of the circuit does not matter, see Lemma 21). Invertibility of $\tilde{f}|_{y=c_j}$ and $\tilde{u_0}|_{y=c_j}$ follows from the fact that we chose c_j's randomly. In particular, $\tilde{f}(\overline{0}, y)$, and so $\tilde{u_0}(\overline{0}, y)$, have roots in $\mathbb{F}$ which are distinct from c_j, $j \in [d_0]$. Thus, $\tilde{f}(\overline{x}, c_j)$ and $\tilde{u_0}(\overline{x}, c_j)$ have non-zero constants and so are invertible in $\mathbb{F}[[\overline{x}]]$. Similarly, $\gamma_\ell/(c_j - g'_{\ell,\delta-1})$ exists in $\mathbb{F}[[\overline{x}]]$.

Thus, the matrix recurrence allows us to calculate the polynomials $C_{i,\delta}$ and $D_{i,\delta}$, given their $\delta-1$ analogues, by adding poly(d_0) many wires and nodes. The precomputations costed us size poly(s,δ). Hence, both $C_{i,\delta}$ and $D_{i,\delta}$ has poly(s,δ,d_0) sized circuit.

We can assume we have only one division gate at the top, as for each gate G we can keep track of numerator and denominator of the rational function computed at G, and simulate all the algebraic operations easily in this representation. When we reach precision $\delta = d_0$, we can eliminate the division gate at the top. As D_{i,d_0} is a unit, we can compute its inverse using the power series inverse formula and approximate only up to degree d_0 (Lemma 19). Finally, the circuit for the polynomial $g_i^{\leq d_0} \equiv C_{i,d_0}/D_{i,d_0} \bmod I^{d_0+1}$, for all $i \in [d_0]$, has size poly(s, d_0).

Altogether, it implies that any factor of u_1 has a circuit of size poly(s, d_0). □

4.2 Low Degree Factors of General Circuits: Proof of Theorem 2

Here, we introduce an approach to handle the general case when rad(f) has exponential degree. We show that allowing a special kind of modular division gate gives a small circuit for any low degree factor of f.

The *modular division* problem is to show that if f/g has a representative in $\mathbb{F}[[\overline{x}]]$, where polynomials f and g can be computed by a circuit of size s, then $f/g \bmod \langle \overline{x}^d \rangle$ can be computed by a circuit of size poly(sd). Note that if g is invertible in $\mathbb{F}[[\overline{x}]]$, then the question of modular division can be solved using Strassen's trick of division elimination [68]. But, in our case g is not invertible in $\mathbb{F}[[\overline{x}]]$ (though f/g is well-defined).

Proof of Theorem 2. As discussed before, to show size bound for an arbitrary factor (with low degree) of f, it is enough to show the size bound for the approximations of power series roots. From Theorem 4, $\tilde{f}(\overline{x}, y) = f(\tau\overline{x}) = k \cdot \prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) := \mu_i$ being distinct.

Fix an i from now on. To calculate $g_i^{\leq\delta}$, we iteratively use Newton iteration with multiplicity (as described in Section 1.3) for $\log\delta + 1$ many times. We know that there are rational functions $\hat{g}_{i,t}$ such that $\hat{g}_{i,t+1} := \hat{g}_{i,t} - \gamma_i \cdot \frac{\tilde{f}}{\partial_y \tilde{f}}\Big|_{y=\hat{g}_{i,t}}$ and $\hat{g}_{i,t} \equiv g_i \bmod \langle \overline{x} \rangle^{2^t}$. We compute $\hat{g}_{i,t}$'s incrementally, $0 \leq t \leq \log\delta + 1$, by a circuit with division gates. As before, $\tilde{f}$ and $\partial_y \tilde{f}$ have poly(s) size circuits.

If $\hat{g}_{i,t}$ has S_t size circuit with division, then $S_{t+1} = S_t + O(1)$. Hence, $\hat{g}_{i,\lg\delta+1}$ has poly$(s, \log\delta)$ size circuit with division.

By keeping track of numerator and denominator of the rational function computed at each gate, we can assume that the only division gate is at the top. As the size of $\hat{g}_{i,\log\delta+1}$ was initially poly$(s, \log\delta)$ with intermediate division gates, it is easy to see that when division gates are pushed at the top, it computes A/B with size of both A and B still poly$(s, \log\delta)$.

Finally, a degree δ polynomial factor $h|f$ will require us to estimate $g_i^{\leq\delta}$ for that many i's. Thus, such a factor has poly$(s\delta)$ size circuit, using a single modular division. □

4.3 Closure of Restricted Complexity Classes: Proof of Theorem 3

This subsection is dedicated towards proving closure results for certain algebraic complexity classes. In fact, for "practical" fields like $\mathbb{Q}, \mathbb{Q}_p$, or $\mathbb{F}_q$ for prime-power q, we give efficient randomized algorithm to output the complete factorization of polynomials belonging to that class (stated as Theorem 15). We use the notation $g \mid\mid f$ to denote that g divides f but g^2 does not divide f. Again, we denote $I := \langle x_1, \ldots, x_n \rangle$

Proof of Theorem 3. There are essentially two parts in the proof. The first part talks only about the existential closure results. In the second part, we discuss the algorithm.

Proof of closure: Given f of degree d, we randomly shift by $\tau : x_i \mapsto x_i + y\alpha_i + \beta_i$. From Theorem 4 we have that $\tilde{f}(\overline{x}, y) := f(\tau\overline{x})$ splits like $\tilde{f} = \prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) =: \mu_i$ being distinct. Here is the detailed size analysis of the factors of polynomials represented by various models of our interest.

Size Analysis for Formula: Suppose f has $n^{O(\log n)}$ size formula. To show size bound for all the factors, it is enough to show that the approximations of the power series roots, i.e. $g_i^{\leq d}$ has size $n^{O(\log n)}$ size formula. This follows from the reduction of factoring to approximations of power series roots.

We differentiate $\tilde{f}$ wrt y, $(\gamma_i - 1)$ many times, so that the multiplicity of the root we want to recover becomes exactly one. The differentiation would keep the size poly$(n^{\log n})$ (Lemma 21). Now, we have $(y - g_i) \mid\mid \tilde{f}^{(\gamma_i - 1)}$ and we can apply classical Newton iteration formula (Section 1.3). For all $0 \leq t \leq \log d + 1$, we compute A_t and B_t such that $A_t/B_t \equiv g_i \bmod I^{2^t}$. Moreover, B_t is invertible in $\mathbb{F}[[\overline{x}]]$ ($\because g_i$ is a simple root of $\tilde{f}^{(\gamma_i-1)}$).

To implement this iteration using the formula model, each time there would be a blow up of d^2. Note that in a formula, there can be many copies of the same variable in the leaf nodes and if we want to feed something in that variable, we have to make equally many copies. That means we may need to make s $(=$ size$(f))$ many copies at each step. One can show that it can be reduced to only d^2 many copies by pre-computing (with blow up at most poly(sd)) all the coefficients $C_0, \ldots, C_d$ wrt y, given the formula of $\tilde{f} =: C_0 + C_1 y + \ldots + C_d y^d$ using interpolation (see [63, Lem.5.3]). Using interpolation, we can convert the formula of $\tilde{f}$ and its derivative to the form $C_0 + C_1 y + \ldots + C_d y^d$. In this modified formula, there are $O(d^2)$ many leaves labelled as y. So in the modified formula of the polynomial $\tilde{f}$ and in its derivative, we are computing and plugging in (for y) d^2 copies of $g_i^{<2^t}$ to get $g_i^{<2^{t+1}}$. This leads to d^2 blow up at each step of the iteration.

As B_t's are invertible, we can keep track of the division gates across iterations and, in the end, eliminate them causing a one-time size blow up of poly(sd) (Lemma 20).

Now, assume that size$(A_t, B_t) \leq S_t$. Then we have $S_{t+1} \leq O(d^2 S_t) + $ poly(sd). Finally, we have $S_{\log d+1} = $ poly$(sd) \cdot d^{2\log d} = $ poly$(n^{\log n})$.

Hence, $g_i^{\leq d} \equiv A_{\log d+1}/B_{\log d+1} \bmod I^{d+1}$ has poly$(n^{\log n})$ size formula, and so does every polynomial factor of f after applying τ^{-1}.

Size Analysis for ABP: This analysis is similar to that of the formula model.

Size Analysis for VNP: Suppose f can be computed by a verifier circuit of size, and witness size, $n^{O(\log n)}$. We call both the verifier circuit size and witness size as size parameter. Now, our given polynomial $\tilde{f}$ has $n^{O(\log n)}$ size parameters. As before, it is enough to show that $g_i^{\leq d}$ has $n^{O(\log n)}$ size parameters.

For the preprocessing (taking $\gamma_i - 1$-th derivative of $\tilde{f}$ wrt y), the blow up in the size parameters is only poly$(n^{\log n})$. Now we analyze the blow up due to classical Newton iteration. We compute A_t and B_t such that $A_t/B_t \equiv g_i \bmod I^{2^t}$. Using the closure properties of VNP (discussed in Section C.1), we see that each time there is a blow up of d^4. The main reason for this blow up is due to the *composition* operation, as we are feeding a polynomial into another polynomial.

Assume that the verifier circuit size$(A_t, B_t) \leq S_t$ and witness size $\leq W_t$. Then we have $S_{t+1} \leq O(d^4 S_t) + $ poly$(n^{\log n})$. So, finally we have $S_{\log d+1} = $ poly$(sd) \cdot d^{4\log d} = $ poly$(n^{\log n})$. It is clear that $g_i^{\leq d} \equiv A_{\log d+1}/B_{\log d+1} \bmod I^{d+1}$ has poly$(n^{\log n})$ size verifer circuit. Same analysis works for W_t and witness size remains $n^{O(\log n)}$. Moreover, we get the corresponding bounds for every polynomial factor of f after applying τ^{-1}.

Remark. Recently in a follow-up paper, Chou, Kumar and Solomon [14] have improved our result on VNP, showing that VNP is closed under factors. Their proof uses the reduction of polynomial factoring to power series root approximation. To avoid division gates, they use the slow variant of Newton iteration (as done in [19, Eqn.5], [57, Lem.4.1]) and use it to compute the circuit of an *approximator* polynomial. An approximator polynomial is a polynomial function of the coefficients (w.r.t one variable, say y) of the circuit that gives the power series roots (w.r.t y) approximated up to certain degree. It can be proved that the approximator polynomial has a small sized circuit. To get the approximate power series roots, one has to compose this circuit with the coefficients of the given polynomial. To finish the proof, use Valiant's lemma [72] showing VNP is closed under composition.

The same idea does not solve the VF and VBP closure under factoring questions as it is not clear if there is an approximator polynomial that has a small sized formula or ABP. If one wants to use the slow Newton iteration iteratively, in each step there would be multiplicative blow-up, as in formula and ABP, we have to make copies of the same computation.

The next claim talks about computing gcd of two polynomials in different models.

Claim 8 (Computing formula gcd). *Given two polynomials $f, g \in \mathbb{F}[\overline{x}]$ of degree d and computed by a formula (resp. ABP) of size s. One can compute a formula (resp. ABP) for gcd(f, g), of size poly$(s, d^{\log d})$, in randomized poly$(s, d^{\log d})$ time.*

Proof of Claim 8. Suppose, gcd$(f, g) =: h$ is of degree $d > 0$, then we will compute $h(\tau\overline{x})$ for a random map τ as in Theorem 4. We know wlog that $\tilde{f} := f(\tau\overline{x}) = \prod_i (y - A_i)^{a_i}$ and $\tilde{g} := g(\tau\overline{x}) = \prod_i (y - B_i)^{b_i}$, where $A_i, B_i \in \mathbb{F}[[\overline{x}]]$. Since $\mathbb{F}[\overline{x}] \subset \mathbb{F}[[\overline{x}]]$ are UFDs (Proposition 1), we could say wlog that $h(\tau\overline{x}) = \prod_{i \in S} (y - A_i)^{\min(a_i, b_i)}$, where $S = \{i \mid A_i = B_i\}$ after possible rearrangement. Now, as τ is a random invertible map, we can assume that, for $i \neq j$, $A_i \neq B_j$ and that $A_i(\overline{0}) \neq B_j(\overline{0})$. So, it is enough to compute $A_i^{\leq d}$ and $B_j^{\leq d}$ and compare them using evaluation at $\overline{0}$. If indeed $A_i = B_i$, then $A_i^{\leq d} = B_i^{\leq d}$. If they are not, they mismatch at the constant term itself! Hence, we know the set S and so we are done once we have the power series roots with repetition. Using univariate factoring, wrt y, we get all the multiplicities, of the roots, a_i and b_i's, additionally we get the corresponding starting points of classical Newton iteration, i.e. $A_i(\overline{0})$ and $B_i(\overline{0})$'s.

Size analysis: We compute $A_i^{\leq d}$ and $B_i^{\leq d}$ by NI, (possibly) after making the corresponding multiplicity one by differentiation. A detailed analysis would show that each approximate root has poly$(s, d^{\log d})$ size formula (resp. ABP). This directly implies that gcd$(\tilde{f}, \tilde{g})$ has poly$(s, d^{\log d})$ size formula (resp. ABP). By taking the product of the linear factors, truncating to degree d, and applying τ^{-1}, we can compute the polynomial gcd(f, g).

Randomization is needed for τ and possibly for the univariate factoring over $\mathbb{F}$. Also, it is important to note that $\mathbb{F}$ may not be algebraically closed. Then one has to go to an extension, do the algebraic operations and return back to $\mathbb{F}$. For details, see Section 5.2. □

Randomized Algorithm. We give the broad steps of our algorithm below. We are given $f \in \mathbb{F}[\overline{x}]$, of degree $d > 0$, as input.

(1) Choose $\overline{\alpha}, \overline{\beta} \in_r \mathbb{F}^n$ and apply $\tau : x_i \to x_i + \alpha_i y + \beta_i$. Denote the transformed polynomial $f(\tau\overline{x})$ by $\tilde{f}(\overline{x}, y)$. Wlog, from Theorem 4, $\tilde{f}$ has factorization of the form $\prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$, where $\mu_i := g_i(\overline{0})$ are distinct.
(2) Factorize $\tilde{f}(\overline{0}, y)$ over $\mathbb{F}[y]$. This will give γ_i and μ_i's.
(3) Fix $i = i_0$. Differentiate $\tilde{f}$, wrt y, $(\gamma_{i_0} - 1)$ many times to make g_{i_0} a simple root.
(4) Apply Newton iteration (NI), on the differentiated polynomial, for $k := \lceil \log(2d^2 + 1) \rceil$ iterations; starting with the approximation μ_{i_0} (mod I). We get $g_{i_0}^{<2^k}$ at the end of the process (mod I^{2^k}).
(5) Apply the transformation $x_i \mapsto Tx_i$ (T acts as a degree-counter). Consider $\tilde{g}_{i_0} := g_{i_0}^{<2^k}(T\overline{x})$. Solve the following homogeneous linear system of equations, over $\mathbb{F}[\overline{x}]$, in the unknowns u_{ij} and v_{ij}'s,
$$\sum_{0 \le i+j < d} u_{ij} \cdot y^i T^j \;=\; (y - \tilde{g}_{i_0}) \cdot \sum_{\substack{0 \le i < d \\ 0 \le j < 2^k}} v_{ij} \cdot y^i T^j \mod T^{2^k} .$$
Solve this system, using Lemma 18, to get a nonzero polynomial (if one exists) $u := \sum_{0 \le i+j < d} u_{ij} \cdot y^i T^j$.
(6) If there is no solution, return "f is irreducible".
(7) Otherwise, find the minimal solution wrt $\deg_y(u)$ by brute force (try all possible degrees wrt y; it is in $[d-1]$).
(8) Compute $G(\overline{x}, y, T) := \gcd_y(u(\overline{x}, y, T), \tilde{f}(T\overline{x}, y))$ using Claim 8.
(9) Compute $G(\overline{x}, y, 1)$ and transform it by $\tau^{-1} : x_i \mapsto x_i - \alpha_i y - \beta_i$, $i \in [n]$, and $y \mapsto y$. Output this as an irreducible polynomial factor of f.

The details of correctness and size-bound of the roots can be proved via a series of lemmas and claims stated below (for the proof of those claims, see the full version)

Claim 9 (Existence). *If f is reducible, then the linear system (Step 5) has a non-trivial solution.*

Claim 10 (Step 8's success). *If the linear system (Step 5) has a non-trivial solution, then $0 < \deg_y G \le \deg_y u < d$.*

Next we show that if one takes the minimal solution u (wrt degree of y), then it will correspond to an irreducible factor of f. We will use the same notation as above.

Claim 11 (Irred. factor). *Suppose $y - g_{i_0} \mid \tilde{f}_1$ and f_1 is an irreducible factor of f. Then, $G = c \cdot \tilde{f}_1(Tx, y)$, for $c \in \mathbb{F}^*$, and $\deg_y(G) = \deg_y(u) = \deg_y(f_1)$ in Step 8.*

Alternative to Claim 8: The above proof (Claim 11) suggests that the gcd question of Step 8 is rather special: One can just write u as $\sum_{0 \le i \le d_1} c_i(\overline{x}, T) y^i$ and then compute the polynomial $G = \sum_{0 \le i \le d_1} (c_i / c_{d_1}) \cdot y^i$ as a formula (resp. ABP), by eliminating division (Lemma 19).

Once we have the polynomial G we can fix $T = 1$ and apply τ^{-1} to get back the irreducible polynomial factor f_1 (with power series root g_{i_0}).

The running time analysis of the algorithm is by now routine. A detailed analysis would establish that the above described algorithm is a randomized poly$(n^{\log n})$-time algorithm that outputs $n^{O(\log n)}$ sized factors. □

Remark.

(1) Above results are true for the classes $VBP(s), VF(s), VNP(s)$ for any size function $s = n^{\Omega(\log n)}$.
(2) By using a *reversal* technique [57, Sec.1.1.2] and a modified τ, our size bound can be shown to be poly$(s, d^{\log r})$, where r (resp. d) is the individual-degree (resp. degree) bound of f. So, when r is constant, we get a factor as a poly(s)-size formula (resp. ABP). Oliveira [57] proved the same result for formulas. But, [57] used *slow* Newton iteration and in each iteration the method was different, owing to which the size was poly(s, d^r).

5 EXTENSIONS

5.1 Closure of Approximative Complexity Classes

In this section, we show that all our closure results, under factoring, can be naturally generalized to corresponding approximative algebraic complexity classes.

Definition 12 (Approximative Closure of a Class [7]). *Let C be an algebraic complexity class over field $\mathbb{F}$. A family (f_n) of polynomials from $\mathbb{F}[\overline{x}]$ is in the* class $\overline{C}(\mathbb{F})$ *if there are polynomials $f_{n;i}$ and a function $t : \mathbb{N} \mapsto \mathbb{N}$ such that g_n is in the class C over the field $\mathbb{F}(\epsilon)$ with $g_n(\overline{x}) = f_n(\overline{x}) + \epsilon f_{n;1}(\overline{x}) + \epsilon^2 f_{n;2}(x) + \ldots + \epsilon^{t(n)} f_{n;t(n)}(\overline{x})$.*

The above definition can be used to define closures of classes like VF, VBP, VP, VNP which are denoted as $\overline{\text{VF}}$, $\overline{\text{VBP}}$, $\overline{\text{VP}}$, $\overline{\text{VNP}}$ respectively. In these cases one can assume wlog that the degrees of g_n and $f_{n;i}$ are poly(n).

Following Bürgisser [9]:- Let $K := \mathbb{F}(\epsilon)$ be the rational function field in variable ϵ over the field $\mathbb{F}$. Let R denote the subring of K that consists of rational functions defined in $\epsilon = 0$. Eg. $1/\epsilon \notin R$ but $1/(1+\epsilon) \in R$.

Definition 13. *[9, Defn.3.1] Let $f \in \mathbb{F}[x_1, \ldots, x_n]$. The* approximative complexity $\overline{\text{size}}(f)$ *is the smallest number r, such that there exists F in $R[x_1, \ldots, x_n]$ satisfying $F|_{\epsilon=0} = f$ and circuit size of F over* constants *K is $\le r$.*

Note that the circuit of F may be using division by ϵ implicitly in an intermediate step. So, we cannot simply assign $\epsilon = 0$ and get a circuit free of ϵ. Also, the degree involved can be arbitrarily large wrt ϵ. Thus, potentially $\overline{\text{size}}(f)$ can be smaller than size(f).

Using this new notion of size one can define the analogous class $\overline{\text{VP}}$. It is known to be closed under factors [9, Thm.4.1]. The idea is to work over $\mathbb{F}(\epsilon)$, instead of working over $\mathbb{F}$, and use Newton iteration to approximate power series roots. Note that in the case of $\overline{\mathrm{VF}}$, $\overline{\mathrm{VBP}}$, $\overline{\mathrm{VP}}$ and $\overline{\mathrm{VNP}}$ the polynomials have poly(n) degree. So, by using repeated differentiation, we can assume the power series root (of $\tilde{f} := f(\tau\overline{x})$) to be simple (i.e. multiplicity= 1) and apply classical NI. We need to carefully analyze the implementation of this idea.

Root Finding Using NI over K. For degree-d $f \in \mathbb{F}[\overline{x}]$ if $\overline{\text{size}}(f) = s$ then: $\exists F \in R[\overline{x}]$ with a size s circuit satisfying $F|_{\epsilon=0} = f$. The degree of F wrt $\overline{x}$ may be greater than d. In that case we can extract the part up to degree d and truncate the rest [11, Prop.3.1]. So wlog $\deg_{\overline{x}}(F) = \deg(f)$.

By applying a random τ (using constants $\mathbb{F}$) we can assume that $\tilde{F} := F(\tau\overline{x}) \in R[\overline{x}, y]$ is *monic* (i.e. leading-coefficient, wrt y in $\tilde{F}$, is invertible in R). Otherwise, $\deg_y(\tilde{F}) = \deg_y(\tilde{f}) = \deg_{\overline{x}}(f)$ will decrease on substituting $\epsilon = 0$ contradicting $F|_{\epsilon=0} = f$. Wlog, we can assume that the leading-coefficient of $\tilde{F}$ wrt y is 1 and the y-monomial's degree is d. From now on we have $\tilde{F}|_{\epsilon=0} = \tilde{f}$ and both have their leading-coefficients 1 wrt y.

Let μ be a root of $\tilde{f}(\overline{0}, y)$ of multiplicity one (as discussed before). Define $\hat{F} := \tilde{F}(\overline{x}, y + \mu + \epsilon) - \tilde{F}(\overline{0}, \mu + \epsilon)$. Note that $(\overline{0}, 0)$ is a simple root of $\hat{F}(\overline{x}, y)$ [11, Eqn.5].

So, a power series root y_∞ of $\hat{F}$ can be built iteratively by classic NI (Lemma 22) The above process, when combined with the first part of the proof of Theorem 3, does imply:

Theorem 14 (Approximative factors). *The approximative complexity classes $\overline{\mathrm{VF}}(n^{\log n})$, $\overline{\mathrm{VBP}}(n^{\log n})$ and $\overline{\mathrm{VNP}}(n^{\log n})$ are closed under factors.*

It can be seen that the VNP-closure under factoring result of Chou, Kumar and Solomon [14] extends to $\overline{\mathrm{VNP}}$-closure under factoring. The same question for the classes $\overline{\mathrm{VF}}$ and $\overline{\mathrm{VBP}}$ we leave as an open question. (Though, for the respective bounded individual-degree polynomials we have the result as before.)

5.2 When Field $\mathbb{F}$ is Not Algebraically Closed

We show that all our results "partially" hold true for fields $\mathbb{F}$ which are not algebraically closed. The common technique used in all the proofs is the structural result (Theorem 4) which talks about power series roots with respect to y. Let $E \subsetneq \overline{\mathbb{F}}$ be the smallest field where a root μ_1 can be found (μ_1 is a root of the polynomial after applying random τ and substituting $\overline{x} = \overline{0}$. Say, $g | \tilde{f_1}(\overline{0}, y)$ is the minimal polynomial for μ_1. The degree of the extension $E := \mathbb{F}[z]/(g(z))$ is at most d. So, computations over E can be done efficiently. The key idea is to view $E/\mathbb{F}$ as a vector space and simulate the arithmetic operations over E by operations over $\mathbb{F}$. The details of this kind of simulation can be seen in [74]. Once we have found all the power series roots of $\tilde{f}(\overline{x}, y)$ over $E[[\overline{x}]]$, say starting from each of the conjugates $\mu_1, \ldots, \mu_i \in E$, it is easy to get a polynomial factor in $E[\overline{x}, y]$. This factor will not be in $\mathbb{F}[\overline{x}, y]$, unless E is a splitting field of $\tilde{f_1}(\overline{0}, y)$. A more practical method is: While solving the linear system over E in Steps 5-7 (Algorithm in Theorem 3) we can demand an $\mathbb{F}$-solution u. Basically, at the level of algorithm in Lemma 18, we can rewrite the linear system $Mw = (\sum_{0 \leq i \leq d} M_i z^i) \cdot w = 0$ as $M_i w = 0$ $(i \in [0, d])$, where the entries of the matrix M_i are given as formulas (resp. ABP) computing a poly(n) degree polynomial in $\mathbb{F}[\overline{x}]$. This way we get the desired $\mathbb{F}$-solution u. Then, Steps 8-9 will yield an irreducible polynomial factor of f in $\mathbb{F}[\overline{x}, y]$. This sketches the following more practical version of Theorem 3.

Theorem 15. *For $\mathbb{F}$ a number field, a local field, or a finite field (with characteristic $> \deg(f)$), there exists a randomized poly$(sn^{\log n})$-time algorithm that: for a given $n^{O(\log n)}$ size formula (resp. ABP) f of poly(n)-degree and bitsize s, outputs $n^{O(\log n)}$ sized formulas (resp. ABPs) corresponding to each of the nontrivial factors of f.*

Note that over these fields there are famous randomized algorithms to factor univariate polynomials in the base case, see [74, Part III] & [60].

The allRootsNI method in Theorem 1 seems to require all the roots $\mu_i, i \in [d_0]$, to begin with. Let $\tilde{u}_1 := \text{rad}(u_1(\tau\overline{x}))$. Since μ_i's are in the *splitting field* $E \subset \overline{\mathbb{F}}$ of $\text{rad}(\tilde{u}_1(\overline{0}, y))$, we do indeed get the size bound of the power series roots $g_i^{\leq d_0}$ of $\tilde{u}_1$ assuming the constants from E. As seen in the proof, any irreducible polynomial factor $\tilde{h}_i := h_i(\tau\overline{x})$ of $\text{rad}(\tilde{u}_1)$ is some product of these $(y - g_i^{\leq d_0})$'s mod I^{d_0+1}. So, for the polynomial $\tilde{h}_i$ in $\mathbb{F}[\overline{x}, y]$ we get a size upper bound over constants E. We leave it as an open question to transfer it over constants $\mathbb{F}$ (note: $E/\mathbb{F}$ can be of exponential degree).

5.3 Multiplicity Issue in Prime Characteristic

The main obstruction in prime characteristic is when the multiplicity of a factor is a p-multiple, where $p \geq 2$ is the characteristic of $\mathbb{F}$. In this case, all versions of Newton iteration fail. This is because the derivative of a p-powered polynomial vanishes. When p is greater than the degree of the input polynomial, these problems do not occur, so all our theorems hold (also see Section 5.2).

When p is smaller than the degree of the input polynomial in Theorem 3, adapting an idea from [45, Sec.3.1], we claim that we can give $n^{O(\lambda \log n)}$-sized formula (resp. ABP) for the p^{e_i}-th power of f_i, where f_i is a factor of f whose multiplicity is divisible exactly by p^{e_i}, and λ is the number of distinct p-powers that appear.

Note that presently it is an open question to show that: If a circuit (resp. formula resp. ABP) of size s computes f^p, then f has a poly(sp)-sized circuit (resp. formula resp. ABP).

Theorem 3 can be extended to all characteristic as follows.

Theorem 16. *Let $\mathbb{F}$ be of characteristic $p \geq 2$. Suppose the poly(n)-degree polynomial given by a $n^{O(\log n)}$ size formula (resp. ABP) factors into irreducibles as $f(\overline{x}) = \prod_i f_i^{p^{e_i} j_i}$, where $p \nmid j_i$. Let $\lambda := \#\{e_i | i\}$.*

Then, there is a poly$(n^{\lambda \log n})$-size formula (resp. ABP) computing $f_i^{p^{e_i}}$ over $\overline{\mathbb{F}}_p$.

High Degree Case. Note that the above idea cannot be implemented efficiently in the case of high degree circuits. Still we can extend our Theorem 1 using allRootsNI. The key observation is that the allRootsNI formula still holds but the summands that appear are exactly the ones corresponding to g_i with $\gamma_i \neq 0 \bmod p$.

This motivates the definition of a partial radical: $\text{rad}_p(f) := \prod_{p \nmid e_i} f_i$, if the prime factorization of f is $\prod_i f_i^{e_i}$.

Theorem 17. *Let $\mathbb{F}$ be of characteristic $p \geq 2$. Let $f = u_0 u_1$ such that size(f)+size$(u_0) \leq s$. Any factor of $\text{rad}_p(u_1)$ has size poly$(s + \deg(\text{rad}_p(u_1)))$ over $\overline{\mathbb{F}}$.*

Proof idea: Observe that the roots with multiplicity divisible by p do not contribute to the allRootsNI process. So, the process works with $\mathrm{rad}_p(u_1)$ and the linear algebra complexity involved is polynomial in its degree.

6 CONCLUSION

The old *Factors conjecture* states that for a nonzero polynomial f: $g \mid f \implies \mathrm{size}(g) \leq \mathrm{poly}(\mathrm{size}(f), \deg(g))$. Motivated by Theorem 1, we would like to strengthen it to:

Conjecture 1 (radical). *For a non-zero polynomial f, the following relation holds: $\min\{\deg(\mathrm{rad}(f)), \mathrm{size}(\mathrm{rad}(f))\} \leq \mathrm{poly}(\mathrm{size}(f))$.*

Is the Radical conjecture true if we replace size by $\overline{\text{size}}$?

In low degree regime also there are many open questions. Can we identify a class "below" VP that is closed under factoring? We conclude with some interesting questions.

(1) Are VF, VBP closed under factoring? We might consider Theorem 3 as a positive evidence. Additionally, note that these classes are already closed under e-th root taking. This is easy to see using the classic Taylor series of $(1+f)^{1/e}$, where $f \in \langle \overline{x} \rangle$.
Can we show closure results for the classes which are contained in $VF(n^{\log n})$ but larger than VF? For example, is $\mathrm{VF}(n^{\log\log n})$ closed under factoring?
(2) Can we find a suitable analog of Strassen's (non-unit) division elimination for high degree circuits? This, by Theorem 2, will resolve Factors conjecture.
(3) Our results weaken when $\mathbb{F}$ is not algebraically closed or has a small prime characteristic (Sections 5.2, 5.3). Can we strengthen the methods to work for all $\mathbb{F}$?

A PRELIMINARIES

This section is intended for preliminaries, most of the claims will be stated without giving detailed proof or proof idea.

A.1 Randomized Algorithm for Linear Algebra Using PIT

The following lemma is an adapted version from [45] discusses how to perform linear algebra when the coefficients of vectors are given as formula (resp. ABP). This will be crucially used in Theorem 3 when we would give an algorithm to output the factors.

Lemma 18. *(Linear algebra using PIT [45, Lem.2.6]) Let $M = (M_{i,j})_{k\times n}$ be a matrix (where k is $n^{O(1)}$) with each entry being a degree $\leq n^{O(1)}$ polynomial in $\mathbb{F}[\overline{x}]$. Suppose, we have algebraic formula (resp. ABP) of size $\leq n^{O(\log n)}$ computing each entry. Then, there is a randomized $\mathrm{poly}(n^{\log n})$-time algorithm that either:*

- *finds a formula (resp. ABP) of size $\mathrm{poly}(n^{\log n})$ computing a non-zero $u \in (\mathbb{F}[\overline{x}])^n$ such that $Mu = 0$, or*
- *outputs 0 which declares that $u = 0$ is the only solution.*

A.2 Basic Operations on Formula, ABP and Circuit

We use the following standard results on size bounds for performing some basic operations (like taking derivative) of circuits, formulas, ABPs.

Lemma 19. *(Eliminate single division [68], [66, Thm.2.1]) Let f and g be two degree-D polynomials, each computed by a circuit (resp. ABP resp. formula) of size-s with $g(\overline{0}) \neq 0$. Then $f/g \bmod \langle \overline{x} \rangle^{d+1}$ can be computed by $O((s+d)d^3)$ (resp. $O(sd^2D)$ resp. $O(sd^2D^2)$) size circuit (resp. ABP resp. formula).*

Proof Idea. Assume wlog that $g(\overline{0}) = 1$. Using the identity, $f/g = f/(1-(1-g)) = f + f(1-g) + f(1-g)^2 + f(1-g)^3 + \cdots$., and truncation using Strassen's *homogenization* trick, in the case of circuits and ABPs (see [63, Lem.5.2]), and an *interpolation* trick in the case of formulas (which also works for ABPs and low degree circuits, [63, Lem.5.4]). A careful analysis shows that the size blow up is at most $O((s+d)d^2 \cdot d)$ (resp. $O(sd \cdot D \cdot d)$ resp. $O(sd \cdot D^2 \cdot d)$) for circuits (resp. ABP resp. formula).

Using the above result, it is easy to see, that we get $\mathrm{poly}(s,d)$ size circuit (resp. ABP resp. formula) for computing $f/g \bmod \langle \overline{x} \rangle^{d+1}$. □

Remark. Note that it may happen that $g(\overline{0}) = 0$. In such a case, We can shift the polynomials f, g by some random $\overline{\alpha} \in \mathbb{F}^n$ and compute $f(\overline{x}+\overline{\alpha})/g(\overline{x}+\overline{\alpha})$ using the method described above. Finally, we recover the polynomial f/g by applying the reverse shift $\overline{x} \mapsto \overline{x} - \overline{\alpha}$.

What if our model has several division gates?

Lemma 20. *(Div. gates elimination [66, Thm.2.12]) Let f be a polynomial computed by a circuit (resp. formula), using division gates, of size s. Then, $f \bmod \langle \overline{x} \rangle^{d+1}$ can be computed by $\mathrm{poly}(sd)$ size circuit (resp. formula).*

Lemma 21 (Derivative computation). *If a polynomial $f(\overline{x}, y)$ can be computed by a circuit (resp. formula resp. ABP) of size s and degree d. Then, any $\frac{\partial^k f}{\partial y^k}$ can be computed by circuit (resp. formula resp. ABP) of size $\mathrm{poly}(sk)$.*

Proof. The idea is simply to use the homogenization and interpolation properties [63, Sec.5.1-2] when the polynomial is of degree $d \leq \mathrm{poly}(s)$. When degree is higher, [35, Thm.1] shows that $\frac{\partial^k f}{\partial y^k}$ can be computed by a circuit of size $O(k^2 s)$. □

B USEFUL IN SECTION 3

Lemma 22. *(Power series root [13, Thm.2.31]) Let $P(\overline{x}, y) \in \mathbb{F}(\overline{x})[y]$, $P'(\overline{x}, y) = \frac{\partial P(\overline{x},y)}{\partial y}$ and $\mu \in \mathbb{F}$ be such that $P(\overline{0}, \mu) = 0$ but $P'(\overline{0}, \mu) \neq 0$. Then, there is a unique power series S such that $S(\overline{0}) = \mu$ and $P(\overline{x}, S) = 0$ i.e.*

$$y - S(\overline{x}) \mid P(\overline{x}, y).$$

Moreover, there exists a rational function y_t, $\forall t \geq 0$, such that

$$y_{t+1} = y_t - \frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)} \text{ and } S \equiv y_t \bmod \langle \overline{x} \rangle^{2^t} \text{ with } y_0 = \mu.$$

Proof Idea. We can inductively prove existence and uniqueness together. Suppose $P = \sum_{i=0}^{d} c_i y^i$. We show that there is y_t, a rational function $\frac{A_t}{B_t}$ such that $y_t \in \mathbb{F}[[\overline{x}]]$, For all $t \geq 0$, $P(\overline{x}, y_t) \equiv 0 \bmod \langle \overline{x} \rangle^{2^t}$ and for all $t \geq 1$, $y_t \equiv y_{t-1} \bmod \langle \overline{x} \rangle^{2^{t-1}}$. The proof is by induction. Let $y_0 := \mu$. Thus, base case is true. Now suppose such y_t exists. Define $y_{t+1} := y_t - \frac{P(\overline{x},y_t)}{P'(\overline{x},y_t)}$. We use Taylor expansion to show that $P(\overline{x},y_{t+1}) = 0 \mod \langle\overline{x}\rangle^{2^{t+1}}$.

Moreover, using the notion of limit, we have unique S, a power series such that $\lim_{t\to\infty} y_t = S$, a formal power series. In particular, we get that $P(\overline{x},S) = 0$ or $y - S \mid P$. □

Lemma 23 (Transform to monic). *For a polynomial $f(\overline{x})$ of total degree $d \geq 0$ and random $\alpha_i \in_r \mathbb{F}$, the transformed polynomial $g(\overline{x},y) := f(\overline{\alpha}y + \overline{x})$ has a nonzero constant as coefficient of y^d, and degree wrt y is d.*

Proof. Suppose the transformation is $x_i \mapsto x_i + \alpha_i y$ where $i \in [n]$. Write $f = \sum_{|\overline{\beta}|=d} c_{\overline{\beta}}\overline{x}^{\overline{\beta}} +$ lower degree terms . Coefficient of y^d in g is $\sum_{|\overline{\beta}|=d} c_{\overline{\beta}}\overline{\alpha}^{\overline{\beta}}$. Clearly, for a random $\overline{\alpha}$ this coefficient will not vanish [65], and it is the highest degree monomial in g.

This ensures $\deg_y(g) = \deg(f) = d$ and that g is monic wrt y. □

C USEFUL IN SECTION 4

For the detailed proofs of the following lemmas, see the full version.

Lemma 24 (Matrix inverse). *Let $\mu_i, i \in [d]$, be distinct nonzero elements in $\mathbb{F}$. Define a $d \times d$ matrix A with the (i,j)-th entry $1/(y_i - \mu_j)^2$. Its entries are in the function field $\mathbb{F}(\overline{y})$. Then, det(A) $\neq 0$.*

Proof Idea. The idea is to consider the power series of the function $1/(y_i - \mu_j)^2$ and show that a monomial appears nontrivially in that of det(A) using Vandermonde determinant. □

Remark. If the characteristic of $\mathbb{F}$ is a prime $p \geq 2$ then the above proof needs a slight modification. One should consider the coefficient of $\prod_{i\in[d]} y_i^{s_i-1}$ in det(A) for a set $S = \{s_1, \ldots, s_d\}$ of distinct non-negative integers that are not divisible by p. Moreover, one has to consider 'random' μ_i's to deduce det(A) $\neq 0$.

Lemma 25 (Series inverse). *Let $\delta \geq 1$. Assume that A is a polynomial of degree $< \delta$ and B is a homogeneous polynomial of degree δ, such that $A(\overline{0}) =: \mu \neq 0$. Then, we have the following identity: $\frac{1}{y-(A+B)} \equiv \frac{1}{y-A} + \frac{B}{(y-\mu)^2} \mod \langle\overline{x}\rangle^{\delta+1}$*

Proof Idea. This is based on simple taylor series expansion: $(a-x)^{-1} = 1/a \cdot (\sum_{i\geq 0}(\frac{x}{a})^i)$ □

C.1 Closure Properties for VNP

VNP-size parameter (w,v) of F refers to w being the witness size and v being the size of the verifier circuit f.

Let $F(\overline{x},y), G(\overline{x},y), H(\overline{x})$ have verifier polynomials f,g,h and the VNP size parameters $(w_f,v_f),(w_g,v_g),(w_h,v_h)$ respectively. Let the degree of F wrt y be d. Then, the following closure properties can be shown ([13] or [12, Thm.2.19]):

(1) Add (resp. Multiply): $F+G$ (resp. FG) has VNP-size parameter $(w_f + w_g, v_f + v_g + 3)$.
(2) Coefficient: $F_i(\overline{x})$ has VNP-size parameter $(w_f, (d+1)(v_f + 1))$, where $F(\overline{x},y) =: \sum_{i=0}^{d} F_i(\overline{x})y^i$.
(3) Compose: $F(\overline{x},H(\overline{x}))$ has VNP-size parameter $((d+1)(w_f + dw_h), (d+1)^2(v_f + v_h + 1))$.

ACKNOWLEDGMENTS

We thank Rafael Oliveira for extensive discussions regarding his works and about circuit factoring in general. In particular, we used his suggestions about VNP and $\overline{\text{VP}}$ in our results. We are grateful to the organizers of WACT'16 (Tel Aviv, Israel) and Dagstuhl'16 (Germany) for the stimulating workshops. P.D. would like to thank CSE, IIT Kanpur for the hospitality and Google India Research Program Team for the support of all travel expenses. N.S. thanks the funding support from DST (DST/SJF/MSA-01/2013-14). A.S. would like to thank Microsoft Research Lab India for partially supporting the travel expenses. We thank Manindra Agrawal, Sumanta Ghosh, Partha Mukhopadhyay, Thomas Thierauf and Nikhil Balaji for the discussions.

REFERENCES

[1] Oliver Aberth. 1973. Iteration methods for finding all zeros of a polynomial simultaneously. *Mathematics of computation* 27, 122 (1973), 339–344.
[2] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. 2017. *Bootstrapping variables in algebraic circuits.* Technical Report. https://www.cse.iitk.ac.in/users/nitin/research.html. (To appear in 50th ACM Symposium on Theory of Computing (STOC), 2018).
[3] Manindra Agrawal and V Vinay. 2008. Arithmetic circuits: A chasm at depth four. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on.* IEEE, 67–75.
[4] Eric Allender and Fengming Wang. 2011. On the power of algebraic branching programs of width two. *Automata, Languages and Programming* (2011), 736–747.
[5] Michael Ben-Or and Richard Cleve. 1992. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.* 21, 1 (1992), 54–58.
[6] Lenore Blum, Mike Shub, and Steve Smale. 1989. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society* 21, 1 (1989), 1–46.
[7] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. 2017. On Algebraic Branching Programs of Small Width. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia.* 20:1–20:31.
[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning.* ACM, 89–96.
[9] Peter Bürgisser. 2001. The complexity of factors of multivariate polynomials. In *In Proc. 42th IEEE Symp. on Foundations of Comp. Science.*
[10] Peter Bürgisser. 2001. On implications between P-NP-hypotheses: Decision versus computation in algebraic complexity. In *MFCS.* Springer, 3–17.
[11] Peter Bürgisser. 2004. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics* 4, 4 (2004), 369–396. (Preliminary version in FOCS 2001).
[12] Peter Bürgisser. 2013. *Completeness and reduction in algebraic complexity theory.* Vol. 7. Springer Science & Business Media.
[13] Peter Bürgisser, Michael Clausen, and Amin Shokrollahi. 2013. *Algebraic complexity theory.* Vol. 315. Springer Science & Business Media.
[14] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. 2018. Some Closure Results for Polynomial Factorization and Applications. ECCC Report TR18-052. In *Electronic Colloquium on Computational Complexity (ECCC).*
[15] Richard Courant, Herbert Robbins, and Ian Stewart. 1996. *What is Mathematics?: an elementary approach to ideas and methods.* Oxford University Press, USA.
[16] Germund Dahlquist and Åke Björck. 2008. Numerical methods in scientific computing, volume I. *Society for Industrial and Applied Mathematics* (2008).
[17] Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. 2014. Homomorphism Polynomials Complete for VP. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS.* 493–504.
[18] Émile Durand. 1960. Solutions numériques des équations algébriques. Tome I, Équations du type F(x)= 0, racines d'un polynôme. (1960).
[19] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. 2009. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.* 39, 4 (2009), 1279–1293. (Preliminary version in STOC'08).
[20] Louis W Ehrlich. 1967. A modified Newton method for polynomials. *Commun. ACM* 10, 2 (1967), 107–108.
[21] Michael A Forbes and Amir Shpilka. 2015. Complexity Theory Column 88: Challenges in Polynomial Factorization. *ACM SIGACT News* 46, 4 (2015), 32–49.

[22] Michael A. Forbes and Amir Shpilka. 2017. A PSPACE Construction of a Hitting Set for the Closure of Small Algebraic Circuits. *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), 163. (To appear in 50th ACM Symposium on Theory of Computing (STOC), 2018).

[23] Michael A Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. 2016. Proof complexity lower bounds from algebraic circuit complexity. In *Proceedings of the 31st Conference on Computational Complexity*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 32.

[24] Patrizia Gianni, Barry Trager, and Gail Zacharias. 1988. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation* 6, 2 (1988), 149–167.

[25] Philip E Gill, Walter Murray, Michael A Saunders, John A Tomlin, and Margaret H Wright. 1986. On projected Newton barrier methods for linear programming and an equivalence to KarmarkarâĂŹs projective method. *Mathematical programming* 36, 2 (1986), 183–209.

[26] Joshua A Grochow. 2015. Unifying known lower bounds via geometric complexity theory. *computational complexity* 24, 2 (2015), 393–475.

[27] Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. 2016. Boundaries of VP and VNP. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Vol. 55. 34:1–34:14.

[28] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. 2018. Algebraic dependencies and PSPACE algorithms in approximative complexity. *arXiv preprint arXiv:1801.09275* (2018).

[29] Venkatesan Guruswami and Madhu Sudan. 1998. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*. IEEE, 28–37.

[30] Gábor Ivanyos, Marek Karpinski, Lajos Rónyai, and Nitin Saxena. 2012. Trading GRH for algebra: algorithms for factoring polynomials and related structures. *Math. Comp.* 81, 277 (2012), 493–531.

[31] Maurice J Jansen. 2011. Extracting Roots of Arithmetic Circuits by Adapting Numerical Methods.. In *2nd Symposium on Innovations in Computer Science (ICS 2011)*. 87–100. http://homepages.inf.ed.ac.uk/mjansen1/Jansen10.pdf

[32] Valentine Kabanets and Russell Impagliazzo. 2003. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 355–364.

[33] Erich Kaltofen. 1985. Computing with Polynomials Given by Straight-Line Programs I: Greatest Common Divisors. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. 131–142.

[34] Erich Kaltofen. 1986. Uniform Closure Properties of P-Computable Functions. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. 330–337.

[35] Erich Kaltofen. 1987. Single-factor Hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 443–452.

[36] Erich Kaltofen. 1989. Factorization of polynomials given by straight-line programs. *Randomness and Computation* 5 (1989), 375–412.

[37] Erich Kaltofen. 1990. Polynomial factorization 1982-1986. *Dept. of Comp. Sci. Report* (1990), 86–19.

[38] Erich Kaltofen. 1992. Polynomial factorization 1987–1991. *LATIN'92* (1992), 294–313.

[39] Erich Kaltofen and Pascal Koiran. 2008. Expressing a fraction of two determinants as a determinant. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*. ACM, 141–146.

[40] Zohar S Karnin and Amir Shpilka. 2009. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*. IEEE, 274–285.

[41] Neeraj Kayal. 2011. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1409–1421.

[42] Neeraj Kayal and Nitin Saxena. 2006. Complexity of ring morphism problems. *computational complexity* 15, 4 (2006), 342–390.

[43] Gregor Kemper. 2010. *A course in Commutative Algebra*. Vol. 256. Springer Science & Business Media.

[44] Immo O Kerner. 1966. Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. *Numer. Math.* 8, 3 (1966), 290–294.

[45] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. 2015. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity* 24, 2 (2015), 295–331.

[46] Steven G Krantz and Harold R Parks. 2012. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

[47] Mrinal Kumar and Shubhangi Saraf. 2016. Arithmetic Circuits with Locally Low Algebraic Rank. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*. 34:1–34:27.

[48] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*. ACM, 296–303.

[49] Grégoire Lecerf. 2002. Quadratic Newton iteration for systems with multiplicity. *Foundations of Computational Mathematics* 2, 3 (2002), 247–293.

[50] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. 1990. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 564–572.

[51] Richard J Lipton and Larry J Stockmeyer. 1978. Evaluation of polynomials with super-preconditioning. *J. Comput. System Sci.* 16, 2 (1978), 124–139.

[52] Meena Mahajan. 2014. Algebraic complexity classes. In *Perspectives in Computational Complexity*. Springer, 51–75.

[53] Ketan Mulmuley. 2017. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society* 30, 1 (2017), 225–309.

[54] Ketan D. Mulmuley. 2012. The GCT Program Toward the P vs. NP Problem. *Commun. ACM* 55, 6 (June 2012), 98–107.

[55] Ketan D. Mulmuley. 2012. Geometric Complexity Theory V: Equivalence between Blackbox Derandomization of Polynomial Identity Testing and Derandomization of Noether's Normalization Lemma. In *FOCS*. 629–638.

[56] Isaac Newton. 1669. De analysi per aequationes numero terminorum infinitas [On analysis by infinite series] (in Latin). (1669). (published in 1711 by William Jones).

[57] Rafael Oliveira. 2016. Factors of low individual degree polynomials. *Computational Complexity* 2, 25 (2016), 507–561. (Preliminary version in CCC'15).

[58] James M Ortega and Werner C Rheinboldt. 2000. *Iterative solution of nonlinear equations in several variables*. SIAM.

[59] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. 2016. Algebraic Independence over Positive Characteristic: New Criterion and Applications to Locally Low Algebraic Rank Circuits. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*. 74:1–74:15. (In print, Computational Complexity, 2018).

[60] Sebastian Pauli. 2001. Factoring polynomials over local fields. *Journal of Symbolic Computation* 32, 5 (2001), 533–547.

[61] David Alan Plaisted. 1977. New NP-hard and NP-complete polynomial and integer divisibility problems. In *Foundations of Computer Science, 18th Annual Symposium on*. IEEE, 241–253.

[62] David Alan Plaisted. 1977. Sparse complex polynomials and polynomial reducibility. *J. Comput. System Sci.* 14, 2 (1977), 210–221.

[63] Ramprasad Saptharishi. 2016. A survey of lower bounds in arithmetic circuit complexity. *URL https://github. com/dasarpmar/lowerbounds-survey/releases. Version* 3, 0 (2016).

[64] Claus-Peter Schnorr. 1977. Improved lower bounds on the number of multiplications/divisions which are necessary to evaluate polynomials. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 135–147.

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

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

[67] Gaurav Sinha. 2016. Reconstruction of Real Depth-3 Circuits with Top Fan-In 2. In *31st Conference on Computational Complexity*.

[68] Volker Strassen. 1973. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik* 264 (1973), 184–202.

[69] Madhu Sudan. 1997. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of complexity* 13, 1 (1997), 180–193.

[70] Brook Taylor. 1715. Methodus Incrementorum Directa et Inversa [Direct and Reverse Methods of Incrementation] (in Latin). (1715). (Translated into English in Struik, D. J. (1969). A Source Book in Mathematics 1200âĂŞ1800. Cambridge, Massachusetts: Harvard University Press. pp. 329âĂŞ332.).

[71] Leslie G. Valiant. 1979. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*. 249–261.

[72] Leslie G. Valiant. 1982. Reducibility by algebraic projections. *L'Enseignement Mathematique: Logic and Algorithmic, Geneva* 2 (1982), 365–380.

[73] Leslie G. Valiant, Sven Skyum, Stuart Berkowitz, and Charles Rackoff. 1983. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.* 12, 4 (1983), 641–644.

[74] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern computer algebra*. Cambridge university press.

[75] Joachim von zur Gathen and Erich Kaltofen. 1985. Factoring sparse multivariate polynomials. *J. Comput. System Sci.* 31, 2 (1985), 265–287.

[76] Oscar Zariski and Pierre Samuel. 1975. *Commutative algebra. II. Reprint of the 1960 edition*. Vol. 29. Graduate Texts in Mathematics.

[77] Hans Zassenhaus. 1969. On Hensel factorization, I. *Journal of Number Theory* 1, 3 (1969), 291–311.