Discovering the roots: Uniform closure results for algebraic classes under factoring *

PRANJAL DUTTA, Chennai Mathematical Institute (& IIT Kanpur)
NITIN SAXENA, Indian Institute of Technology, Kanpur
AMIT SINHABABU, Aalen University, Germany

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 \( 2^{Ω(n)} \)-hard polynomial then any nonzero multiple \( \prod f_i^{e_i} \) 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 (respectively algebraic branching programs) is closed under factoring. We show that given a polynomial \( f \) of degree \( n^{O(1)} \) and formula (respectively ABP) size \( n^{O(\log n)} \) we can find a similar size formula (respectively ABP) factor in randomized \( n^{O(\log n)} \)-time. Consequently, if determinant requires \( n^{Ω(\log n)} \) size formula, then the same can be said about any of its nonzero multiples.

In all our proofs, we exploit the following property of multivariate polynomial factorization. Under a random linear transformation \( \tau \), the polynomial \( f(\tau x) \) 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 vast body of classical results and concepts in algebraic circuit factorization (eg. Zassenhaus, J.NT 1969; Kaltofen, STOC 1985-7 & Bürgisser, FOCS 2001).

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

Additional Key Words and Phrases: circuit factoring, formula, ABP, randomized, hard, VF, VBP, VP, VNP, quasipoly

ACM Reference Format:

* A preliminary version was presented in STOC 2018.

Authors’ addresses: Pranjal Dutta, pranjal@cmi.ac.in, Chennai Mathematical Institute (& IIT Kanpur); Nitin Saxena, nitin@cse.iitk.ac.in, Indian Institute of Technology, Kanpur; Amit Sinhababu, amitks@cse.iitk.ac.in, Aalen University, Germany.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.
Manuscript submitted to ACM
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 problem both from the point of view of practice and theory. We study the following two questions related to multivariate polynomial factorization:

(1) Closure properties: Let \( \{f_n(x_1, \ldots, x_n)\}_n \) be a polynomial family in an algebraic complexity class \( C \) (e.g., VP, VBP, VNP or \( \overline{V} \) 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) Uniformity: Can we design an efficient, i.e., randomized poly\((n)\)-time, algorithm to output the factor \( g_n \) with a representation in \( C \)?

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 [Mah14, SY10, BCS13].

Algebraic circuits are a natural model to represent polynomials compactly. An algebraic circuit has the structure of a 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 (the length of the longest path in the circuit), formal 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 (respectively VF) contains the families of \( n \)-variate polynomials of degree \( n^{O(1)} \) over \( \mathbb{F} \), computed by \( n^{O(1)} \)-sized circuits (respectively formulas). The class VF is sometimes denoted as \( \mathit{VP}_e \), for it collects ‘expressions’ which is another name for formulas. Similarly, one can define VQP (respectively 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 (respectively 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 \( \text{VP}_{nb} \).

Algebraic branching program (ABP) is another model for computing polynomials, which we define in Section 2.1. The class VBP contains the families of polynomials computed by \( n^{O(1)} \)-sized ABPs. We have the easy containments: \( VF \subseteq VBP \subseteq VQP = VQF \); for details we refer [BOC92, VSBR83].

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 \( n \), \( f_n(\overline{x}) = \sum_{\omega \in \{0,1\}^{t(n)}} g_n(\overline{x}, \omega_1, \ldots, \omega_{t(n)}) \). Here, the witness size is \( t(n) \) and the 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 [Val79]).

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 [Kal90, Kal92, FS15].

1.1 Previously known closure results

Famously, Kaltofen [Kal85a, Kal86, Kal87, Kal89] showed that VP is uniformly closed under factoring, i.e. for a given degree \( d \) degree \( n \) variate polynomial \( f \) of circuit size \( s \), there exists a randomized poly\((s d)\)-time algorithm that outputs its factor as a circuit whose size is bounded by poly\((s d)\). This fundamental result has several applications such as ‘hardness versus randomness’ in algebraic complexity [KI03, AV08, DSY09, AGS19, CKS19b, KST19, DST21, Dut21].
derandomization of Noether Normalization Lemma [Mul17], in the problem of circuit reconstruction [KS09, Sin16], and polynomial equivalence testing [Kay11]. In general, multivariate polynomial factoring has several applications, including decoding of Reed-Solomon, Reed-Muller codes [GS98, Sud97], integer factoring [LLMP90], primary decomposition of polynomial ideals [GTZ88] and algebra isomorphism [KS06, IKRS12]. Typically algorithms for multivariate polynomial factorization use univariate factorization as a subroutine. For factoring univariate polynomials over rationals, Lenstra, Lenstra and Lovasz [LLL82] gave a deterministic polynomial time algorithm. For factoring univariate polynomials over finite fields, there are randomized polynomial time algorithms due to Berlekamp [Ber70], Cantor and Zassenhaus [CZ81]. Even over other fields (complex, p-adics, algebraic number fields) and rings (Galois rings etc), several results on univariate factoring are known [Sch82, Chi94, CG00, Lan85, Len83, vZGH96, DMS19].

It is natural to ask whether Kaltofen’s VP factoring result can be extended to VP_n^e which allows the 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 $x^{2^e} - 1$ can be computed in size $s$, but it has factors (of high degree) over $C$ that require circuit size $\Omega \left( \frac{2^{e^2}}{\sqrt{s}} \right)$ [LS78, Sch77]. It is conjectured [Bürgisser, Conj.8.3] that low degree factors of high degree small-sized circuits have small circuits. Formally, any factor $g$ of a polynomial $f$ computed by an arithmetic circuit of size $s$, can be computed by an arithmetic circuit of size $\text{poly}(s, d_g)$, where $d_g$ is the degree of factor $g$. Kaltofen asked this as an open question [Kal86, Kal87] and Bürgisser posed this as “Factor Conjecture” [Bürgisser]. The Conjecture has several interesting consequences in algebraic complexity. For more exposure, see [Bürgisser], [Bürgisser, Remark 8.15] and the recent survey [Gro19].

Partial results towards the Factor Conjecture are known. Kaltofen [Kal87] showed the following result. If a 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 $\text{poly}(e, \deg(g), s)$. The question left open is to remove the dependency on $e$. In the special case where $f = g^e$, it was proved in [Kal87] that $g$ has circuit size $\text{poly}(\deg(g), \text{size}(f))$. Kaltofen also observed that if $f = g^e h$ and degree of $h$ is $\text{poly}(s)$, then $g$ can be computed by a small circuit.

In the high degree regime, we do not expect uniform closure results, as several algorithmic problems related to factoring are NP-hard there, e.g. computing the degree of the squarefree part, gcd, or lcm. Even in the special case of supersparse (or lacunary) univariate polynomials (represented as a list of nonzero coefficients and the binary encoding of exponents of corresponding terms), the above mentioned problems are NP-hard [Pla77]. Nevertheless, efficient algorithms are known for computing bounded degree factors of supersparse polynomials ([Gre16] and references therein).

Now, we discuss the closure results for classes more restrictive than VP (such as VF, VBP, etc.). Unfortunately, Kaltofen’s technique [Kal89] for VF will give a superpolynomial-sized factor formula; as it heavily reuses intermediate computations in the steps of Hensel lifting, linear system solving, and gcd computation. The same holds for the class VBP. In contrast, extending the idea of [DSY09], Oliveira [Oli16] showed that an $n$-variate polynomial with bounded individual degree and computed by a formula of size $s$, has factors of formula size $\text{poly}(n, s)$. Furthermore, it was established [Oli16] that for a given $n$-variate individual-degree-$r$ polynomial, computed by a circuit (respectively formula) of size $s$ and depth $\Delta$, there exists a $\text{poly}(n', r)$-time randomized algorithm that outputs any factor of $f$ computed by a circuit (respectively formula) of depth $\Delta + 5$ and size $\text{poly}(n', s)$. A special case of factoring algebraic branching programs was...
considered in [KK08]—it dealt with the elimination of a single division gate from skew circuits (also see Section 2.1 & Lemma 5), and another special case was solved in [Jan11].  

Going beyond VP we can ask about the closure of VNP. Bürgisser conjectured [Bür13, Conj.2.1] that VNP is closed under factoring. Kaltofen’s technique [Kal89] for factoring VP circuits does not yield the closure of VNP.  

**Looking at the Border.** Recently, approximative algebraic complexity classes like $\overline{VP}$ [GMQ16] have become objects of interest, especially in the context of the geometric complexity program [Mul12a, Mul12b, Gro15]. A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ has approximative (or border) complexity $\leq s$ if there is an infinite sequence of circuits (using arbitrary elements from the field $\mathbb{F}(\varepsilon)$ as scalars) of size $\leq s$ computing $f_\varepsilon \in \mathbb{F}(\varepsilon)[x_1, \ldots, x_n]$, where $\varepsilon$ is a new variable such that $\lim_{\varepsilon \to 0} f_\varepsilon = f$. For example, if a polynomial $f(\overline{x}, \varepsilon) = e^d g(\overline{x}) + e^{d+1} h(\overline{x}, \varepsilon)$ can be computed by an arithmetic circuit of size $\varepsilon^{-1}$, then $g$ has an approximative circuit of size $s$. Thus, approximative complexity (size) of $g$ must be $\leq s$.

The class $\overline{VP}$ (the closure of VP) contains families of polynomials of degree poly($n$) that can be approximated (infinitesimally closely) by poly($n$)-sized circuits. Bürgisser [Bür04, Bür01] discusses approximative complexity of factors, proving that low degree factors of high degree circuits have small approximative complexity. Thus, the Factor Conjecture is true in the setting of approximative complexity. Also, $\overline{VP}$ is closed under factoring [Bür04, Theorem 4.1]. For the standard algebraic complexity classes, we can ask whether their approximative closure is closed under factors.

We conclude by stating a few reasons why closure results under factoring are interesting and non-trivial. First, some classes are not closed under factors. For example, the family of sparse polynomials (the number of monomials with nonzero coefficients in $f_\varepsilon$ is bounded by $n^{O(1)}$); this is because a factor’s sparsity may blowup super-polynomially [vzGK85]. However, the recent breakthrough work of Bhargava, Saraf and Volkovich [BSV20] gave $s^{\text{poly}(d) \cdot \log n}$ sparsity upper bound for factors of $n$-variate polynomials of individual degree $d$ and sparsity $s$. In particular, for constant individual degree sparse polynomials, the factors can have at most quasi-polynomial blowup in sparsity.

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 useful for proving lower bounds on the power of some algebraic proof systems [FSTW16].

Finally, factoring of arithmetic circuits is crucially used in many hardness versus randomness results in algebraic complexity. Kabanets and Impagliazzo showed that the famous problem of black-box derandomization of polynomial identity testing (PIT) for the class VP can be solved if we can prove strong enough arithmetic circuit lower bounds (get an explicit hard polynomial family); for details see [KI03, Theorem 4.1].

Suppose a polynomial $f(\overline{y})$ is such that for a nonzero size-$s$ circuit $C$, $C(\overline{y}, f(\overline{y})) = 0$. 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 family $\{f_n\}_n$ then $(\overline{y}, f_n(\overline{y}))$ would be a hitting-set generator (hsg) for the circuit family $\{C_n\}_n$ [KI03, Theorem 7.7].

### 1.2 Our results

Before stating the results, we describe some of the assumptions and notations used throughout the paper. Set $[n]$ refers to $\{1, 2, \ldots, n\}$ while $[a, b]$ for $a < b$ and $a, b \in \mathbb{N}$ means for all integers $a \leq i \leq b$. 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 $\mathbb{C}$).

---

2Recently, in [ST20] Sinhababu and Thierauf proved that VBP is closed under factors.

3Recently, in [CKS19b] Chou, Kumar and Solomon confirmed that VNP is indeed closed under factors.
\( \mathbb{R}, \mathbb{Q}, \mathbb{Q}_p \) or finite fields of characteristic \( > \) degree of the input polynomial). For a brief discussion on this issue, see Section 6.

**Ideal.** We denote the variables \((x_1, \ldots, x_n)\) as \( \mathfrak{x} \). The ideal \( I := \langle \mathfrak{x} \rangle \) 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 \( 0 \) [Tay15]).

**Radical.** For a polynomial \( f = \prod_i f_i^{e_i} \), with \( f_i \)'s being coprime irreducible nonconstant factors of multiplicity \( e_i > 0 \), the squarefree part \( \prod_i f_i \) is called as the radical of \( f \), denoted as \( \text{rad}(f) \). The radical of a polynomial is unique (up to scalar).

What can we say about the size of the radical of \( f \), if \( f \) has a circuit of size \( s \)? We prove that the size of the radical is \( \text{poly}(s, d_r) \), where \( d_r \) is the degree of the radical. As a direct corollary, we get that every factor of a given circuit \( C \) has size bounded by a polynomial in size(\( C \)) and the degree of the radical of \( C \). Thus, our main result gives a good circuit size bound for factors when \( \text{rad}(f) \) has small degree.

A more general formulation is:

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

Note that Kaltofen’s proof technique in [Kal89] does not extend to the exponential degree regime (even when the degree of \( \text{rad}(f) \) 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. We do not know how to extend the proof technique in Kaltofen’s single factor Hensel lifting paper [Kal87, Theorem 2] that works for the perfect-power case of \( f = g^\epsilon \). It can be seen that \( \text{rad}(f) \) equals \( f/\gcd(f, \partial_{x_i}(f)) \), but the \( \gcd \) itself can be of exponential degree and so one cannot hope to use [Kal87, Theorem 4] to compute the \( \gcd \) either. It is an open question whether the \( \gcd \) of two polynomials (computed by small circuits of high degree) can be computed by a small circuit [Kal87].

**Remarks.** (1) Interestingly, our result when combined with [Kal87, Theorem 3] implies that every factor \( g \) of \( f \) has a circuit of size polynomial in: \( \text{size}(f) \), \( \text{deg}(g) \) and \( \min(\{\deg(\text{rad}(f)), \text{size}(\text{rad}(f))\}) \). We leave it as an open question whether the latter expression is polynomially related to \( \text{size}(f) \).

(2) Theorem 1 shows an interesting way to create hard polynomials. In the theorem statement let the size concluded be \( (s + \deg(\text{rad}(u_1)))^c \), for some constant \( c \). If one has a polynomial \( f_i \) with \( x_1, \ldots, x_n \) that is \( 2^{cn} \)-hard, then any nonzero \( f := \prod_i f_i^{e_i} \) is also \( 2^{O(n)} \)-hard for arbitrary positive \( e_i \)'s, as long as \( \sum_i \deg(f_i) \leq 2^{cn/2} - 1 \).

1.2.1 A detour into numerical analysis (via arithmetic circuits). Root approximation of univariate polynomials has been an interesting problem in mathematics & engineering. Interestingly, it has also found applications in various other problems such as computing the largest eigenvalue, and checking whether a matrix is approximately PSD (positive semi-definite) [LV16].

We can quantify the same question about ‘approximating roots’ of a univariate polynomial and analyze the bit-complexity of the root; where the measure is the bitsize of the best circuit. For an \( f(x) \in \mathbb{R}[x] \) we define bitsize(\( f \)) = \( s \) if we can compute \( f(x) \) by a circuit using \( \{+, -, \times, \div\} \) gates and of overall bitsize \( s \). Bit-size of a constant \( c \) is the number of bits (binary) required to represent \( c \) (which is \( \log_2(c) \)). Note that, this is in contrast with the usual VP notion where constants are free. Here, the degree of \( f \) can be exponential (eg. \( 2^n \)) and the coefficients may have exponential bit-length (eg. \( 2^n \) bits, or values \( 2^{-2^n} \) to \( 2^{2^n} \)).

For this complexity notion, we can show a surprising fact for the roots.
Claim 1. For each root \( a \in (0, 1) \) of \( f(x) \), there is some \( 2^m \)-bit approximation \( a' \) such that \text{bitsize}(a') \leq O((s+m)\cdot \log(\frac{1}{\epsilon})) \), where \text{bitsize}(f) \approx s \text{ and } \epsilon \in (0, 1) \) lower bounds the gap between \( a \) and the other roots of \( f(x) \).

A proof is sketched in the appendix (Section A).

Note that the estimate is nontrivial, as it is essentially expressing \( 2^m \) bits of the root using 'only' bitsize \( m \); so, roots of small circuits are rather special, in contrast to generic strings that are incompressible \([VL97]\). It is relevant here to recall Shub-Smale's tau conjecture that states that 'small' circuits have 'few' integral roots! A proof of tau conjecture would imply \( \text{P} \neq \text{NP} \) over \( \mathbb{C} \) \([BCSS98]\). This motivates well the study of the roots of circuits.

Our Theorem 1 is an algebraic analog of the above; there \( m \log(\frac{1}{\epsilon}) \) gets replaced by the degree of the radical (or, the number of roots). Also, the algebraic result is better (than Theorem 1) in the sense that we do not need \( \div \) gates.

1.2.2 Back to multivariate algebraic models. In general, for a high degree circuit \( f \), \( \text{rad}(f) \) can be of high degree (exponential in the size of the circuit). Ideally, we would like to show that every degree \( d \) factor of \( f \) has \( \text{poly}(\text{size}(f), d) \)-size circuit. The next theorem reduces the above question to a special kind of modular division, where the denominator polynomial may \emph{not} be invertible but the quotient is well-defined (eg. \( x^2/x \mod x \)). All that remains is to eliminate somehow this kind of non-unit division operator (which we leave as an open question). Consider random \( \{ \} \) elements \( a_i, \beta_i \in \mathbb{F} \) and the corresponding random linear map \( \tau : x_i \mapsto a_iy + x_i + \beta_i, i \in [n] \), where \( y \) is a new variable apart from \( x_1, \ldots, x_n \). In the following theorem, the notation \( A/B \mod (\mathbb{F})^{d+1} \) means we are truncating the power series \( C = A/B \) upto total degree \( d \) (the lower part).

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

Note that in Theorem 2, \( B \) may be non-invertible in \( \mathbb{F}[\mathbb{X}]/(\mathbb{F})^{d+1} \) and may have a high degree (eg. \( 2^d \)). So, we cannot use the famous trick of Strassen to do division elimination here \([Str73]\).

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

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

Moreover, there exists a randomized \( \text{poly}(n^{\log n}) \)-time algorithm that: for a given \( n^{O(\log n)} \) sized formula (respectively ABP) \( f \) of \( \text{poly}(n) \)-degree, outputs \( n^{O(\log n)} \) sized formula (respectively 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 \([BSS89]\). In the former case, our algorithm takes \( \text{poly}(n^{\log n}) \) bit operations (assuming that the characteristic is zero or larger than the degree; see Theorem 26 in Section 6.2).

It is important to note that Theorem 3 does not follow by invoking Kaltofen’s circuit factoring \([Kal89]\) and VSBR transformation \([VSBR83]\) from circuit to log-depth formula. Formally, if we are given a formula (respectively ABP) of

\[\text{By a random choice } \sigma \in \mathbb{F} \text{ we will mean that choose randomly from a fixed finite set } S \subseteq \mathbb{F} \text{ of appropriate size. This will be in the spirit of the Polynomial Identity Lemma (Lemma 9).}\]
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 (respectively ABP), one will 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 efficiently; an excellent overview of the implementation level details to keep in mind is [KSS15, Section 3].

Our proof methods extend to the approximative versions $C(n^{\log n})$ for $C \in \{\overline{VP}, \overline{VBP}, \overline{VNP}\}$ as well (Theorem 25).

As before, Theorem 3 has an interesting lower bound consequence: If $f$ has VF (respectively VBP respectively VNP) complexity $n^{o(\log n)}$, then any nonzero $fg$ has similar hardness (for deg(g) \leq poly(n)). In fact, the method of Theorem 3 yields a formula factor of size $s^d \frac{d^2 \log d}{2}$ 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 $n^{O(\log n)}$ size formula.

1.3 Polynomial factoring and power series roots

All our results use a reduction of polynomial factoring to approximating the power series roots of a polynomial. Here, first, we try to sketch why approximating the power series roots suffice to compute the factors. Subsequently, we sketch a new method called allRootsNI which approximates the roots simultaneously.

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 connect linear factors with 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 linear factor 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 [CRS96, §2.5.4]). So, if we go to the algebraic closure of $\mathbb{F}(x_1, \ldots, x_n)$, any multivariate polynomial which is monic in $x_n$ will split into factors all linear in $x_n$. A representation of the elements of $\mathbb{F}(x_1, \ldots, x_{n-1})$ as a finite circuit is impossible (eg. $\sqrt{x_1}$). On the other hand, all the roots (wrt a new variable $y$) are actually elements from the power series ring $\mathbb{F}[[x_1, \ldots, x_n]]$, after a random linear transformation on the variables, $\tau : \mathbb{F} \mapsto \mathbb{F} + \alpha y + \beta$, is applied (Theorem 17). This is a direct consequence of the classical idea of Newton iteration in the formal power series setting [BSC13, Theorem 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 $\frac{3}{2} x + \frac{3/2 \times 1/2}{2!} x^2 + \ldots$.

Formally, Theorem 17 shows that under a random $\tau : \mathbb{F} \mapsto \mathbb{F} + \alpha y + \beta$, where $\alpha, \beta \in \mathbb{F}$, polynomial $f$ can be factored as $f(\tau X) = \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i}$, where $g_i \in \mathbb{F}[\mathbb{F}]$ with the constant terms $g_i(\tilde{b})$ being distinct, $d_0 := \deg(\gcd(f))$ and $\gamma_i > 0$.

Computing factors of a polynomial can be reduced to approximating power series roots. Theorem 17 implies that any factor $g$ of degree $d$ can be completely split as $\prod_{i=1}^{d_0} (y - g_i)$, where $g_i$ is a power series (which are also roots of $f$). If we know the approximations of $g_i$ up to degree $d$ (notationally; $g_i^d := g_i \bmod (x_1, \ldots, x_n)^{d+1}$), we can compute $g$ exactly.

In particular, suppose $G = \prod_{i=1}^{d_0} (y - g_i^d)$. Observe that the factor $g = G^\leq d$, and thus, we have computed $g$ accurately. For the details, see Section 3.1.
The power series roots can be approximated using the classical Newton iteration method [BCS13, Theorem 2.31], which works when the root we want to compute has multiplicity one. If a factor has multiplicity $e \geq 2$, then all its roots would have multiplicity $e$. If $e$ is $\text{poly}(s)$, then we can take $(e-1)$-th derivative of the polynomial to be factored. In this derivative polynomial (which can be computed by a small circuit), the roots we wanted have multiplicity 1 (and thus one can use the classical Newton iteration, see Lemma 15). If $e$ is exponential in $s$, computing circuits of derivatives of order $e$ may lead to an exponential blow-up of size [Kal87]. In that case, we devise a new method that approximates all the roots simultaneously. If the number of roots is $\text{poly}(s)$ (equivalently, the degree of the radical is $\text{poly}(s)$), then our method shows that factors have small circuits. We briefly sketch the overall idea in the next paragraph.

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

Define, $\hat{f}(x, y) := f(x) = \prod_i (y - g_i)^{y_i}$. Applying the derivative operator $\frac{\partial}{\partial y}$ on $\hat{f}$, we get a classic identity (which we call logarithmic derivative identity\footnote{Very recently, this identity has found applications in other contexts, eg. constant top-fanin (6-bottom-fanin) depth-4 PIT [DDS21b], and restricted de-bordering results in GCT [DDS21a]. Since, it converts the product gate to a sum gate, quite often the expressions become very useful.}): $\frac{\partial}{\partial y} \hat{f} = \sum_{i=1}^{d_0} \frac{y_i}{1 - g_i}$ . Reduce the above identity modulo $I^{\delta+1}$ and define $\mu_i := g_i(0) \equiv g_i \mod I$. This gives us (see Claim 2):

$$\frac{\partial}{\partial y} \hat{f} = \sum_{i=1}^{d_0} \frac{y_i}{1 - g_i} \equiv \sum_{i=1}^{d_0} \frac{y_i}{1 - g_i^{\leq \delta}} + \sum_{i=1}^{d_0} \frac{y_i \cdot g_i^{e, \delta}}{(y - \mu_i)^2} \mod I^{\delta+1}.$$  

In terms of the $d_0$ unknowns $g_i^{e, \delta}$, the above is a linear equation. (Note: We treat $y_i, \mu_i$‘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 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 $M(i, j) := \frac{y_i}{(c_j - \mu_j)^2}$. Vector $v_{\delta}$ respectively $W_{\delta}$ is a $d_0 \times 1$ matrix where each entry is denoted by $v_{\delta}(i) := g_i^{e, \delta}$ respectively $W_{\delta}(i) := \frac{\partial}{\partial y} \hat{f} \bigg|_{y = c_i} - G_{i, \delta}$, where $G_{i, \delta} := \sum_{k=1}^{d_0} k \cdot y_k / (c_i - g_k^{e, \delta})$. We ensure that $\{c_i, \mu_i \mid i \in [d_0]\}$ are distinct, and show that the determinant of $M$ is nonzero (Lemma 19). So, by knowing approximations up to $\delta-1$, we can recover $\delta$-th (and thus up to degree $\delta$ as well) 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 $I^{\delta+1}$.

To see the gory details, see the proof of Theorem 1 in Section 4.1. For our results on restricted models like formulas and ABPs (Theorem 3), we use the classical Newton iteration method (Lemma 15).

**Comparisons with other techniques:** Most of the works on multivariate polynomial factoring use Hensel Lifting (lifting factorization modulo powers of an ideal). Hensel lifting and Newton iteration (lifting roots) are mathematically related, and various versions of them are equivalent. See [VzG84] for comparisons between these two techniques. Nevertheless, for showing closure results in different models, one viewpoint may be more useful than the other. Kaltofen’s classic works mostly used the Hensel lifting viewpoint, although his bivariate factoring [Kal85b] showed either can be used.

Sasaki and Sasaki [SS93] used power series roots for multivariate factoring over different fields, but their way of approximating the roots (via a version of Hensel lifting) and reconstructing factors from roots (via different linear
combinations of powers of roots) are different from us. Our approach of using power series roots is inspired by the approach of Oliveira [Oli16]. There are significant technical differences (in handling the non-monic case and in using different versions of Newton iteration and most importantly, the simultaneous root approximation rather than one at a time), still the core idea of using approximate roots to prove factor size bound is same. Some of the algorithmic ideas (reconstructing a factor by computing a minimal polynomial of an approximate root using linear algebra) can be found in the classic LLL algorithm [LLL82] and the bivariate polynomial factoring algorithm of Kaltofen [Kal85b]. Later, Bürgisser [Bür04] used Newton iteration to approximate a root and then find a minimal polynomial for the root by solving a linear system (the trick of using an additional variable \( t \), performing a substitution \( x_1 \mapsto tx_1 \) and then seeing the polynomials in \( t \) with coefficients over \( \mathbb{F}(\overline{x}) \), helped to extend the bivariate case to general multivariate). Our work uses ideas from some of these prior works.

There are numerical methods (similar to Newton iteration) to simultaneously approximate all the roots of a univariate polynomial [Dur60, Ker66, Ehr67, Abe73]. However, it is well known that the Newton Iteration fails to approximate the roots that repeat (see [Lec02]) in the algebraic setting. The same holds for these techniques as well. In contrast, the allRootsNI technique can handle roots with high multiplicity.

Organization of the article. In the next section, we recall some basic operations for different algebraic models of computation and some necessary algebraic tools which will be useful for us. In Section 3, we discuss the classical Newton iteration formula and the usefulness of power series root approximation to factoring. We prove Theorem 1–2 in Section 4 which focuses on factor size bound of high degree polynomials computed by small sized circuits. Section 5 is devoted to proving Theorem 3, which focuses on factoring (size bound and algorithm) in the quasipolynomial size regime for different algebraic models. Section 6 talks about results for approximative complexity classes and special fields (when it is of low characteristic or it is not algebraically closed). Finally, we conclude with some open questions in Section 7. Appendix A is for a detour and can be seen as a numerical analog (and application) of Theorem 1.

2 PRELIMINARIES

This section is mainly divided into two subsections—Section 2.1 is on the basic definition and operations of different algebraic models which will be used to upper bound the size of factors computed by respective models and exactly compute it at times, while Section 2.2 is for congregating the useful mathematical tools required for our results. Before that, we talk about the fundamental mathematical notion that we use throughout the article, namely power series.

The formal power series ring is denoted as \( \mathbb{F}[\![ x_1, \ldots, x_n ]\!] \). The elements of this ring are multivariate formal power series, with degree as precision. So, an element \( f \) is written as \( f = \sum_{i=0}^{n} f^i \), 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 [Kem10, Chap.13]). The truncation \( f \lesssim d \), 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 \( f \). We will also denote \( f^{<d} \) as degree \( < d \) part of \( f \) i.e. \( f^{<d} := f^{\lesssim d} \). In fact, it is obvious that \( f^{\lesssim d} = f^{<d} + f^{\geq d} \) for all \( d \geq 0 \).

The advantages of the ring \( \mathbb{F}[\![ x ]\!] \) are many. They usually emerge because of the inverse identity:

\[
(1 - x_1)^{-1} = \sum_{i \geq 0} x_1^i
\]

which would not have made sense in \( \mathbb{F}[\![ x ]\!] \) but is available now. This fact will be used in division elimination in different algebraic models (Lemma 5). For fast algorithms for various basic operations on formal power series, we refer the
2.1 A Primer on Algebraic Models

In our proofs, we will need some basic results about formulas, ABPs, and circuits. In particular, we can efficiently eliminate a division gate, extract a homogeneous part, and compute derivatives. For more exposure, see [SY10, Sap19].

A formula is a rooted binary tree where internal nodes are labeled $+$ or $\times$ and leaf nodes are labeled from the set $\mathbb{F}_\cup X$, where $X$ is the set of indeterminates. The size is the number of nodes and edges of the tree.

Algebraic circuits correspond to directed acyclic graphs where each node is a source node (indegree 0) labeled from the set $\mathbb{F}_\cup X$, or has indegree 2 (fanin 2) \(^6\) and is labeled $+$ or $\times$. A designated sink node (outdegree 0) is the output node. Each node computes a polynomial in an obvious way, and the graph computes the polynomial at the output node.

The acyclicity constraint ensures that there is a linear ordering of the nodes such that each node, or instruction, only uses previously computed polynomials. The fanout or the outdegree of any intermediate node can be $> 1$ and thus, the reuse of nodes helps to compute more intricate polynomials in small-sized circuits, which might not be possible in formulas.

ABP is a skew circuit, i.e. each multiplication gate has fanin two with at least one of its inputs being a variable or a field constant. A completely different definition can be given via layered graphs or iterated matrix multiplication or symbolic determinant. It is well-known that they are all equivalent up to polynomial blow up [Mah14].

**Definition 4 (Algebraic Branching Program).** An algebraic branching program (ABP) is a layered graph with a unique source vertex (say $s$) and a unique sink vertex (say $t$). All edges are from layer $i$ to $i + 1$ and each edge is labelled by a linear polynomial. The polynomial computed by the ABP is defined as $f = \sum_{\gamma: s \to t} \text{wt}(\gamma)$, where for every path $\gamma$ from $s$ to $t$, the weight $\text{wt}(\gamma)$ is defined as the product of the labels over the edges forming $\gamma$.

The size of the ABP is defined as the total number of edges in the ABP. Width is the maximum number of vertices in a layer.

Equivalently, one can define $f$ as a product of matrices (of dimension at most the width), each one having linear polynomials as entries. For more details, see [SY10].

Determinant is in VBP and is computable by a $n^{O(\log n)}$ size formula. It is a famous result that the ABP model is the same as symbolic determinant [MV97].

**Computing homogeneous components and coefficients of a polynomial**

Let $f(x, y)$ be polynomial of degree $d$ in variables $y$ and $\bar{x} = (x_1, \ldots, x_n)$. Let $C$ be a formula (respectively circuit or ABP) of size $s$ that computes a polynomial $f$. Write $f$ as a polynomial in $y$, with coefficients from $\mathbb{F}[\bar{x}]$, such that $f(x, y) = \sum_{i=0}^{d} f_i y^i$. Then, the coefficients $f_i(\bar{x})$ have formulas (respectively circuits or ABPs) of size $\text{poly}(s, d)$.

First we replace every variable $x_i$ by $T x_i$ to get a formula computing $P(T) = \sum_{i=0}^{d} P_i T^i$. Now we take $d+1$ many distinct numbers $\alpha_0, \ldots, \alpha_d$ that we substitute for $T$. We create $d + 1$ many arithmetic formulas computing $P(\alpha_0), \ldots, P(\alpha_d)$.

We get the following equation where the leftmost matrix is the Vandermonde matrix.

\[^6\]The original circuit can have unbounded fanin. However, it is not hard to convert the same into fanin 2 with constant blowup in size.
We use the following standard results on size bounds for performing some basic operations (eg. division elimination, \(O\) ABPs and low degree circuits); see subsection 2.1 above. A careful analysis shows that the size blow-up is at most computing \(f\) trick, in the case of circuits and an discard the remaining terms of degree greater than \(d\). Note that this doesn’t involve the actual degree of the original circuit \(\Phi\). We briefly give an idea of how to construct a new circuit \(\Psi\) of the claimed functionality:

For every gate \(v\) in \(\Phi\), we define \(d+1\) gates in \(\Phi\) which we denote \((v, 0), \ldots, (v, d)\) in such a way that \((v, i)\) computes the \(i\)-th degree homogeneous part computed at \(v\)-th node in \(\Phi\); polynomial computed at \(v\) in \(\Phi\), call it \(\Phi_v\) and at \((v, i)\)-th gate in \(\Psi\), we call it \(\Psi_{(v,i)}\). We construct \(\Psi\) inductively as follows. If \(v\) is an input gate (variables or constant), we can clearly define \((v, i)\) as an input gate with the appropriate properties. If \(\Phi_v = \Phi_u \times \Phi_w\), define \(\Psi_{(v,i)} = \Psi_{(u,i)} + \Psi_{(w,i)}\) for all \(i\). If \(\Phi_v = \Phi_u \times \Phi_w\), define \(\Psi_{(v,i)} = \sum_{j=0}^{i} \Psi_{(u,j)} \times \Psi_{(w,i-j)}\). Induction implies that \(\Psi\) computes each homogeneous part of \(\Phi\), gate by gate. Every gate in \(\Phi\) corresponds to at most \(O((d+1)^2)\) gates in \(\Psi\) (each product gate requires \(O((d+1)^2)\) additional sum gates), and so size of \(\Psi\) is \(O(s \cdot d^2)\).

This does not work for formulas. We refer to [SY10, Theorem 2.2] and [Sap19, Lemma 5.2] for more details.

Basic operations on formulas, circuits and ABPs

We use the following standard results on size bounds for performing some basic operations (eg. division elimination, taking derivatives) of circuits, formulas, ABPs.

**Lemma 5.** (Eliminate single division [Str73], [SY10, Theorem 2.1]) Let \(f\) and \(g\) be two degree-\(D\) polynomials, each computed by a circuit (respectively ABP respectively formula) of size-\(s\) with \(g(\overline{0})\neq0\). Then \(f \div g \mod (\overline{\mathbb{T}})^{d+1}\) can be computed by \(O((s+d)^3)\) (respectively \(O(sd^2 \cdot D)\) respectively \(O(sd^2 \cdot D^2)\)) size circuit (respectively ABP respectively formula).

**Proof.** Assume wlog that \(g(\overline{0}) = 1\); we can ensure this by appropriate normalization. So, we have the following power series identity in \(\mathbb{F}[\overline{\mathbb{T}}]\):

\[
\frac{f}{g} = \frac{f}{1 - (1 - g)} = f + f \cdot (1 - g) + f \cdot (1 - g)^2 + f \cdot (1 - g)^3 + \cdots.
\]

Note that this is a valid identity as \(1 - g\) is constant free. For all \(d \geq 0\), \(\text{LHS-\text{RHS mod} } (\overline{\mathbb{T}})^{d+1}\).

If we want to compute \(f \div g \mod (\overline{\mathbb{T}})^{d+1}\), we can take the RHS of the above identity up to the term \(f \cdot (1 - g)^d\) and discard the remaining terms of degree greater than \(d\). The degree-\(d\) monomials can be truncated, using Strassen’s homogenization trick, in the case of circuits and an interpolation trick in the case of formulas (which also works for ABPs and low degree circuits); see subsection 2.1 above. A careful analysis shows that the size blow-up is at most \(O((s+d)2^d \cdot d)\) (respectively \(O(sd \cdot D \cdot d)\) respectively \(O(sd \cdot D^2 \cdot d)\)) for circuits (respectively ABP respectively formula).

It is easy to see that using the above result, we get \(\text{poly} \langle s, d \rangle\) size circuit (respectively ABP respectively formula) for computing \(f \div g \mod (\overline{\mathbb{T}})^{d+1}\). \(\square\)
Remark. Note that it may happen that \( g(\overline{0}) = 0 \), thus \( 1/g \) does not exist in \( \mathbb{F}[\overline{x}] \), yet \( f/g \) may be a polynomial of degree \( d \). In such a case, we need to discuss a modified normalization that works. We can shift the polynomials \( f, g \) by some random \( \overline{z} \in \mathbb{F}^d \). The constant term of the shifted polynomial is nonzero with high probability (Lemma 8). Now, we compute \( f(\overline{x} + \overline{z})/g(\overline{x} + \overline{z}) \) using the method described above. Finally, we recover the polynomial \( f/g \) by applying the reverse shift \( \overline{x} \mapsto \overline{x} - \overline{z} \).

What if our model has several division gates?

Lemma 6. (Div. gates elimination [SY10, Theorem 2.12]) Let \( f \) be a polynomial computed by a circuit (respectively formula), using division gates, of size \( s \). Then, \( f \mod (\overline{x})^{d+1} \) can be computed by \( \text{poly}(sd) \) size circuit (respectively formula).

Proof idea. We pre-process the circuit (respectively formula) so that the only division gate used in the modified circuit (respectively formula) is at the top. Now to remove the single division gate at the top, we use the above power series trick.

The idea of the pre-processing is the following. We can separately keep track of numerator and denominator computed at each gate and simulate addition, multiplication and division gates in the original circuit and incrementally go from bottom to top. In particular, if at the \( i \)-th depth we have + gate coming from two nodes at the \( i+1 \)-th depth computing \( u_1/v_1 \) and \( u_2/v_2 \) respectively, then at that specific node it computes \( u_1/v_1 + u_2/v_2 = (u_1 \cdot v_2 + u_2 \cdot v_1)/(v_1 \cdot v_2) \). Similarly, for \( \times \) gate, it is \( (u_1 \cdot u_2)/(v_1 \cdot v_2) \). Note that, given \( u_1, v_1 \) and \( u_2, v_2 \), it only incurs constantly many additional gates.

Thus in a whole, this pre-processing incurs only \( O(s) \) additional blow up in the case of circuits. In the case of formulas one has to ensure that in any path from the leaf to the root, there are only \( O(\log sd) \) division gates. \( \square \)

Lemma 7 (Derivative computation). If a polynomial \( f(\overline{x}, y) \) can be computed by a circuit (respectively formula respectively ABP) of size \( s \) and degree \( d \). Then, any \( \partial f/\partial y \) can be computed by circuit (respectively formula respectively ABP) of size \( \text{poly}(sk) \).

Proof. The idea is to use the homogenization and interpolation properties [Sap19, Section 5.1-2].

Let \( f(\overline{x}, y) = c_0 + c_1 y + c_2 y^2 + \cdots + c_\delta y^\delta \), where \( c_0, c_1, \ldots, c_\delta \in \mathbb{F}[\overline{x}] \). Given the circuit (respectively formula) computing polynomial \( f(\overline{x}, y) \), we can get the circuits (respectively formula respectively ABP) computing \( c_0, \ldots, c_\delta \) using homogenization and interpolation as discussed before. Given \( c_0, \ldots, c_\delta \), computing \( \partial f/\partial y \) in size \( \text{poly}(sd) \) is trivial. We use this approach of computing derivative when the polynomial is of degree \( d \leq \text{poly}(s) \).

Remark. In the case of high degree circuits, we cannot use the above approach. [Kal87, Theorem 1] shows that \( \partial f/\partial y \) can be computed by a circuit of size \( O(k^2 s) \), i.e. the degree of the circuit does not matter. The main idea is to use the Leibniz product rule of \( k \)-th order derivative inductively. Kaltofen [Kal87] gave evidence that it is unlikely that we can compute \( \partial f/\partial y \) in circuit size \( \text{poly}(s, \log k) \). Otherwise, the permanent polynomial would have circuits of polynomial size.

Closure properties for VNP

We now discuss some closure properties of the class VNP which will be crucially required for proving Theorem 3.

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 \( f, g \) and \( h \) with 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 [(BCS13) or [Bü13, Theorem 2.19]]:
(1) Add (respectively Multiply): $F + G$ (respectively $FG$) 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 $((d+1)(w_f + dw_h), (d+1)^2(v_f + v_h + 1))$.

Proof. All the above statements are easy to prove using the definition of VNP.

\[
(FG)(\overline{x}, y) = \left( \sum_{u \in \{0,1\}^{w_f}} f(\overline{x}, u_1, \ldots, u_{w_f}) \right) \cdot \left( \sum_{u \in \{0,1\}^{w_g}} g(\overline{x}, u_1, \ldots, u_{w_g}) \right)
\]

where,

\[
A(\overline{x}, u_1, \ldots, u_{w_f+1}, \ldots, u_{w_f+v_g}) = f(\overline{x}, u_1, \ldots, u_{w_f}) \cdot g(\overline{x}, u_{w_f+1}, \ldots, u_{w_f+v_g})
\]

Trivially, $A$ has size $v_f + v_g + 3$ (extra: one node, two edges) and witness size is $w_f + w_g$. Similarly, with $F + G$.

(2) Interpolation gives, $f_i(\overline{x}) = \sum_{j=0}^{d} \alpha_j F(\overline{x}, \beta_j)$, for some distinct arguments $\beta_j \in \mathbb{F}$. Clearly, $F(\overline{x}, \beta_j)$ has VNP-size parameter $(w_f, v_f)$. Using the previous addition property we get that the verifier circuit has size $(d+1)(v_f + 1)$.

Witness size remains $w_f$ as we can reuse the witness string of $F$.

(3) Write $F(\overline{x}, y) = \sum_{i=0}^{d} F_i(\overline{x})y^i$. We know that $F_i$ has VNP-size parameter $(w_f, (d+1)(v_f + 1))$. For $0 \leq i \leq d$, $H_i$ has VNP-size parameter $(iw_h, (i+1)v_h)$ using $i$-fold product (Item 1). Substituting $y = H$ in $F$, we can calculate the VNP-size parameter.

Suppose $F_i$ and $H_i$ have corresponding verifier circuits $A_i$ and $B_i$ respectively. Then,

\[
F(\overline{x}, H(\overline{x})) = \sum_{i=0}^{d} F_i(\overline{x})H^i(\overline{x})
\]

Thus, the witness size is $< (d+1)(w_f + dw_h)$. The corresponding verifier circuit size is $< (d+1)^2(v_f + v_h + 1)$.

\[\square\]

2.2 Mathematical Toolkit

This section is dedicated for assembling the mathematical tools which we will need throughout.

In the following part, we use the DeMillo-Lipton-Schwartz-Zippel lemma, now called the Polynomial Identity Lemma, which basically shows that a nonzero polynomial evaluates to nonzero at a random point from a large enough field. See [CKS19b] and references therein for more details and the history of this lemma.

**Lemma 8 (Polynomial Identity Lemma [Ore22, DL78, Zip79, Sch80]).** Let $p(x_1, \ldots, x_n)$ be an $n$-variate nonzero polynomial of total degree $d$. Let $S \subseteq \mathbb{F}$ be a finite set. For $\overline{x} \in S^n$ picked independently and uniformly at random,

\[
Pr[p(\overline{x}) = 0] \leq \frac{d}{|S|}.
\]

Remark. The above lemma implies that PIT $\in \text{co-RP}$, i.e. there is an efficient polynomial time randomized algorithm to test whether a given polynomial is zero or not.
By a random choice \( \alpha \in \mathbb{F} \), we always mean that we choose uniformly at random from a fixed finite set \( S \subseteq \mathbb{F} \) of large enough size (say, \( \geq 2d \) if \( d \) is the degree of the nonzero polynomial where we substitute \( \overline{\alpha} \in S^n \)). We will use properties of \( \gcd(f, g) \) and a related determinant polynomial called resultant.

**Sylvester matrix & resultant.** First, let us look at the notion of the resultant of two univariate polynomials. Let \( p(x), q(x) \in \mathbb{F}[x] \) be of degree \( a, b \) respectively. From Euclid’s extended algorithm, it can be shown that there exist two polynomials \( u(x), v(x) \in \mathbb{F}[x] \) such that \( u(x)p(x) + v(x)q(x) = \gcd(p(x), q(x)) \). This is known as Bezout’s identity. If \( \gcd(p(x), q(x)) = 1 \), then \( (u, v) \) with \( \deg(u) < b \) and \( \deg(v) < a \) is unique. Let \( u(x) = u_0 + u_1x + u_2x^2 + \ldots + u_{b-1}x^{b-1} \) and \( v(x) = v_0 + v_1x + \ldots + v_{a-1}x^{a-1} \).

Now, if we use the equation \( u(x)p(x) + v(x)q(x) = \gcd(p(x), q(x)) \) and compare the coefficients of \( x^i \), for \( 0 \leq i < a+b \), we get a system of linear equations in the \( a+b \) many unknowns \( (u_i, v_i) \). The system of linear equations can be represented in the matrix form as \( Mx = y \), where \( x \) consists of the unknowns. The resultant of \( f, g \) is defined as the determinant of the matrix \( M \). It is easy to see that \( M \) is invertible if and only if the polynomials are coprime.

Now, the notion of resultants can be extended to multivariate, by defining the resultant of polynomials \( f(\overline{x}, y) \) and \( g(\overline{x}, y) \) wrt some variable \( y \). The idea is the same as before; now we take \( \gcd \) wrt the variable \( y \) and get a system of linear equations from Bezout’s identity. The matrix can be explicitly written with entries being polynomial coefficients (or they could be from \( \mathbb{F}[\overline{x}] \)). This is known as the Sylvester matrix, which we define next.

**Definition 9.** Let \( f(\overline{x}, y) = \sum_{i=0}^{l} f_i(\overline{x})y^i \) and \( g(\overline{x}, y) = \sum_{i=0}^{m} g_i(\overline{x})y^i \). Define Sylvester matrix of \( f \) and \( g \) wrt \( y \) as the following \((m+l) \times (m+l) \) matrix:

\[
\text{Syl}_{y}(f, g) := \begin{pmatrix}
 f_0 & 0 & \cdots & 0 & g_0 & 0 & \cdots & 0 \\
 f_1 & f_0 & \cdots & 0 & g_1 & g_0 & \cdots & 0 \\
 f_2 & f_1 & \vdots & \vdots & g_1 & \vdots & \vdots & \vdots \\
 \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 f_l & f_{l-1} & f_1 & 0 & g_m & \cdots & 0 \\
 0 & f_l & f_2 & 0 & 0 & g_l & \cdots & 0 \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 0 & 0 & \cdots & f_l & 0 & 0 & \cdots & g_m
\end{pmatrix}.
\]

Now, the resultant can be formally defined as follows (for more details and alternate definitions, see [LN97, Chap.1]).

**Definition 10.** Given two polynomials \( f(\overline{x}, y) \) and \( g(\overline{x}, y) \), define the resultant of \( f \) and \( g \) wrt \( y \) as the determinant of the Sylvester matrix,

\[
\text{Res}_y(f, g) := \det(\text{Syl}_y(f, g)).
\]

From the definition, it can be seen that \( \text{Res}_y(f, g) \) is a polynomial in \( \mathbb{F}[x] \) with degree bounded by \( 2\deg(f)\deg(g) \).

Now, we state the following fundamental property of the Resultant, which is crucially used.

**Proposition 11 (Resultant vs. GCD).**

1. Let \( f, g \in \mathbb{F}[\overline{x}, y] \) be polynomials with positive degree in \( y \). Then, \( \text{Res}_y(f, g) = 0 \iff f \) and \( g \) have a common factor in \( \mathbb{F}[\overline{x}, y] \), which has positive degree in \( y \).
2. There exists \( u, v \in \mathbb{F}[\overline{x}, y] \) such that \( uf + vg = \text{Res}_y(f, g) \).
The proof of this standard proposition can be found in many standard books on algebra including [vzGG13, Section 6]. In the following lemma, by coprimality wrt $y$, we mean that there is no common factor that has a monomial involving the variable $y$. For example, $x^2 y$ and its derivative wrt $y$ (which is $x^2$) are coprime wrt $y$.

**Lemma 12 (Squarefree-ness).** Let $f \in \mathbb{F}([x])[y]$ be a polynomial with $\deg_y (f) \geq 1$. $f$ is squarefree iff $f$ and $f' := \partial_y f$ are coprime wrt $y$.

**Proof.** The main idea is to show that there does not exist $g \in \mathbb{F}([x])[y]$ with positive degree in $y$ such that $g \mid \gcd_y (f(x, y), f'(x, y))$. This is true because—suppose $g$ is an irreducible polynomial with positive degree in $y$ that divides both $f(x, y)$ and $f'(x, y)$.

$$f(x, y) = gh \implies f'(x, y) = gh' + g'h \implies g \mid h'.$$

As $g$ is irreducible and $\deg_y (g') < \deg_y (g)$ we deduce that $g \mid h$. Hence, $g^2 \mid f$. This contradicts the hypothesis that $f$ is squarefree. \qed

Now, we state another standard lemma, which is useful to us and proven using the property of Resultant.

**Lemma 13 (Coprimality).** Let $f, g \in \mathbb{F}([x])[y]$ be coprime polynomials wrt $y$ (i.e. nontrivial in $y$). Then, for $\bar{p} \in \mathbb{F}^n$, $f(\bar{p}, y)$ and $g(\bar{p}, y)$ are coprime (i.e. nontrivial in $y$).

**Proof.** Consider $f = \sum_{i=1}^d f_i y^i$ and $g = \sum_{i=1}^d g_i y^i$. Choose a random $\bar{p} \in \mathbb{F}^n$. Then, by Proposition 11 & Lemma 8, $f_d \cdot g \cdot \text{Res}_y (f, g)$ at $x = \bar{p}$ is nonzero. This, in particular, implies that $\text{Res}_y (f(\bar{p}, y), g(\bar{p}, y)) \neq 0$.

This implies, by Proposition 11, $f(\bar{p}, y)$ and $g(\bar{p}, y)$ are coprime. \qed

**Lemma 14 (Transform to monic).** For a polynomial $f(\bar{x})$ of total degree $d \geq 0$ and random $\alpha_i \in \mathbb{F}$, the transformed polynomial $g(\bar{x}, y) := f(\bar{x} + \alpha y + \bar{\beta})$ has a nonzero constant as the coefficient of $y^d$, and degree wrt $y$ is $d$.

**Proof.** Suppose the transformation is $x_i \mapsto x_i + \alpha_i y + \beta_i$ where $i \in [n]$. Write $f = \sum_{|\bar{p}| = d} c_{\bar{p}} \bar{x}^\bar{p} + \text{lower degree terms}$. The coefficient of $y^d$ in $g$ is $\sum_{|\bar{p}| = d} c_{\bar{p}} \bar{\alpha} \bar{x}^\bar{p}$. Clearly, for a random $\bar{\alpha}$ this coefficient will not vanish (Lemma 8), and it is the highest degree monomial in $g$.

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

**Remark.** In the above statement/proof, we do not need $\bar{\beta}$, in particular, $\bar{\beta} = 0$ works as well. Random $\bar{\beta}$ is required to ensure the existence of different power series roots, see the next section for the details. As we need the more general linear shift later, we use it here as well.

### 3 FACTORIZATION OF POLYNOMIALS OVER POWER SERIES RING

First, we present the proof of classical Newton iteration (in the setting of formal power series). This is also a formal power series version of implicit function theorem [KP12, Sec.1.3].

**Lemma 15.** (Power series root [BCS13, Theorem 2.31]) Let $P(\bar{x}, y) \in \mathbb{F}(\bar{x})[y]$, $P'(\bar{x}, y) = \frac{\partial P(\bar{x}, y)}{\partial y}$ and $\mu \in \mathbb{F}$ be such that $P(\bar{\mu}, 0) = 0$ but $P'(\bar{\mu}, 0) \neq 0$. Then, there is a unique power series $S$ such that $S(\bar{\mu}) = \mu$ and $P(\bar{s}, S) = 0 \text{ i.e. }$

$$y = S(\bar{x}) \mid P(\bar{x}, y).$$

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

$$y_{t+1} = y_t - \frac{P(\bar{x}, y_t)}{P'(\bar{x}, y_t)} \text{ and } S \equiv y_t \mod (\bar{x})^{2^t} \text{ with } y_0 = \mu.$$
Proof. We give an inductive proof of 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 \( y_t \in \mathbb{F}[\{\bar{x}\}] \). For all \( t \geq 0 \), \( P(\bar{x}, y_t) \equiv 0 \mod \langle \bar{x} \rangle^{2t} \) and for all \( t \geq 1 \), \( y_t \equiv y_{t-1} \mod \langle \bar{x} \rangle^{2t-1} \). The proof is by induction. Let \( y_0 := \mu \). Thus, the base case is true. Now suppose such \( y_t \) exists. Define \( y_{t+1} := y_t - \frac{P(\bar{x}, y_t)}{P'(\bar{x}, y_t)} \).

Now, \( y_t \equiv y_{t-1} \mod \langle \bar{x} \rangle^{2t-1} \implies y_t(\bar{0}) = \mu \). Hence \( P'(\bar{x}, y_t)|_{\bar{x}=\bar{0}} = P'(\bar{0}, \mu) \neq 0 \) and so \( P'(\bar{x}, y_t) \) is a unit in the power series ring. So, \( y_{t+1} \in \mathbb{F}[\{\bar{x}\}] \). Let us verify that it is an improved root of \( P \); we use Taylor expansion. For the algebraic version of Taylor expansion, see the treatment in [Bou13].

\[
P(\bar{x}, y_{t+1}) = P\left(\bar{x}, y_t - \frac{P(\bar{x}, y_t)}{P'(\bar{x}, y_t)}\right)
\]

\[
= P(\bar{x}, y_t) - P'(\bar{x}, y_t) \frac{P(\bar{x}, y_t)}{P'(\bar{x}, y_t)} + \frac{P''(\bar{x}, y_t)}{2!} \left( \frac{P(\bar{x}, y_t)}{P'(\bar{x}, y_t)} \right)^2 + \cdots
\]

\[
\equiv 0 \mod \langle \bar{x} \rangle^{2t+1}.
\]

Thus, \( P(\bar{x}, y_{t+1}) \equiv 0 \mod \langle \bar{x} \rangle^{2t+1} \) and \( y_{t+1} \equiv y_t \mod \langle \bar{x} \rangle^{2t} \). This completes the induction step.

Moreover, it is not hard to see that there exists a unique power series \( S \) such that \( S \equiv y_t \mod \langle \bar{x} \rangle^{2t} \) for all \( t \geq 0 \). It is unique as \( \mu \) is a non-repeated root of \( P(\bar{0}, y) \). As it holds for all \( t \geq 0 \), we must have \( P(\bar{x}, S) = 0 \), as otherwise \( P(\bar{x}, S) \not\equiv 0 \mod \langle \bar{x} \rangle^{2t} \) for some \( t \geq 1 \). This, in particular, would imply that \( P(\bar{x}, y_t) \not\equiv 0 \mod \langle \bar{x} \rangle^{2t} \) which is a contradiction! Thus, \( P(\bar{x}, S) = 0 \iff y - S \mid P \). \( \square \)

Remark. In a more general situation, where we have a system of \( n \) polynomials or power series in several variables \( z_1, \ldots, z_n, x_1, \ldots, x_m \), we can compute the power series roots \( g_1(\bar{x}), \ldots, g_n(\bar{x}) \) using a multidimensional version of Newton iteration using the Jacobian. See [Bou13] for details. Also, note that there is a slow version of Newton iteration, which is \( y_{t+1} = y_t - \frac{P(\bar{x}, y_t)}{\partial_y P(\bar{x}, y_t)} \). To approximate roots up to degree \( d \), we have to use this version of Newton iteration \( d \) many times repeatedly. For restricted models, we show that the fast version of NI has crucial advantage over the slow version. There are applications of power series roots/implicit function theorem paradigm in algebraic complexity [DSY09, KS16, PSS16, BJ18], in coding theory [AP00, NRS17, BSCI17], in polynomial system solving [GHM+98, Kri02].

Now, to get the factorization over \( \mathbb{F}[\bar{x}] \), we look into the analytic factorization pattern of a polynomial over the power series ring \( \mathbb{F}[\{x_1, \ldots, x_n\}] \). We need the notion of uniqueness of factorization, which the following classic proposition ensures.

Proposition 16. [ZS75, Chap.VII] The power series ring \( \mathbb{F}[\{x_1, \ldots, x_n\}] \) is a unique factorization domain (UFD), and so is \( \mathbb{F}[\{\bar{x}\}][y] \).

Now, we show that by applying a random linear map, any polynomial splits completely over the ring \( \mathbb{F}[\{\bar{x}\}] \). (Recall: \( \mathbb{F} \) is algebraically closed.)

Theorem 17 (Power Series Complete Split). Let \( f \in \mathbb{F}[\bar{x}] \) with \( \deg(\text{rad}(f)) = d_0 > 0 \). Consider \( \alpha_i, \beta_i \in \mathbb{F} \) 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 \( \mathbb{F}[\{\bar{x}\}] \), \( f(\bar{\tau}(\bar{x})) = k \cdot \prod_{i \in [d_1]} (y - g_i)^{p_i} \), where \( k \in \mathbb{F}^*, y_1 > 0 \), and \( g_i(\bar{0}) = \mu_i \). Moreover, \( \mu_i \)'s are distinct nonzero field elements.

Proof. Let the complete irreducible factorization of \( f \) be \( \prod_{i \in [d_1]} f_i^{p_i} \). We apply a random \( \tau \) so that \( f \), and thus all its factors, become monic in \( y \) (Lemma 14). The monic factors \( \tilde{f}_i := f_i(\tau \bar{x}) \) remain irreducible (because \( \tau \) is invertible). Also,
\( f_i(\overline{0}, y) = f_i(\overline{x} y + \overline{\beta}) \) and \( \partial_y f_i(\overline{0}, y) \) remain coprime (because \( \overline{\beta} \) is random, apply Lemma 13). In other words, \( f_i(\overline{0}, y) \) is squarefree (Lemma 12).

In particular, one can write \( \tilde{f}_i(\overline{0}, y) \) as \( \prod_{j=1}^{\deg(f_i)} (y - \mu_{i,j}) \) for distinct nonzero field elements \( \mu_{i,j} \) (ignoring the constant, which is the coefficient of the highest degree of \( y \) in \( \tilde{f}_i \)). Using classical Newton iteration (see Lemma 15), one can write \( \tilde{f}_i(\overline{x}, y) \) as a product of power series \( \prod_{j=1}^{\deg(f_i)} (y - g_{i,j}) \), with \( g_{i,j}(\overline{0}) := \mu_{i,j} \). Thus, each \( f_i(\overline{x}) \) can be factored into linear factors in \( \mathbb{F}[[\overline{x}]] \mid [y] \).

As the polynomials \( f_i \) are irreducible and coprime, by Lemma 13, it is clear that \( \tilde{f}_i(\overline{0}, y), i \in [m] \), are mutually coprime. Thus, \( \mu_{i,j} \) are distinct and they are \( \sum_j \deg(f_i) = d_0 \) many. Hence, after proper renaming of the roots \( g_{i,j} \), \( f(\overline{x}) \) can be completely factored as \( \prod_{i \in [m]} f_i(\overline{x})^{e_i} = \prod_{i \in [d_0]} (y - g_i)^{\gamma_i} \), with \( \gamma_i > 0 \) and the field constants \( g_i(\overline{0}) \) being distinct. □

**Corollary 18.** Suppose \( g \) is a polynomial factor of \( f \). As before let \( f(\overline{x}) = \prod_{i \in [m]} f_i(\overline{x})^{e_i} = k \cdot \prod_{i \in [d_0]} (y - g_i)^{\gamma_i} \). As \( g(\overline{x}) \mid f(\overline{x}) \) we deduce that \( g(\overline{x}) = k' \cdot \prod (y - g_i)^{\gamma_i} \) with \( 0 \leq \gamma_i \leq \gamma_i \) and \( k' \in \mathbb{F}[\overline{x}] \). If \( g \) irreducible, then \( \gamma_i \in \{0, 1\} \).

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

### 3.1 Reducing factoring to power series root approximation:

Using the power series complete split (Theorem 17), we show that multivariate polynomial factoring reduces to power series root finding up to a certain precision. Following the above notation \( f \) splits as \( f(\overline{x}) = \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i} \) mod \( I^{t+1} \), where \( I := (x_1, \ldots, x_n) \). Note that there is a one-one correspondence, induced by \( \tau \), between the polynomial factors of \( f \) and \( f(\overline{x}) \) (because \( \tau \) is invertible and \( f \) is \( y \)-free).

Next, we show case by case how to find a polynomial factor of \( f(\overline{x}) \) 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 degrees \( \leq (d - 1) \). So, if we compute the approximations of all the power series roots (wrt \( y \)) up to the 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 16).

**Case 2- Computing a monic non-linear factor:** Assume that a factor \( g \) of total degree \( t \) is of the form \( y^k + c_{k-1} y^{k-1} + \cdots + c_1 y + c_0 \), where for all \( i, c_i \in \mathbb{F}[\overline{x}] \). Now, this factor \( g \) also splits into linear (in \( y \)) factors over \( \mathbb{F}[\overline{x}] \) and obviously, these linear factors are also linear factors of the original polynomial \( f(\overline{x}) \). 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 (as required for Theorem 1), we need not find the combination of the power series roots forming a factor algorithmically. Doing it through brute-force search takes exponential time (\( (\frac{d_0}{d})^k \) choices). Interestingly, using a classical idea (solving a linear system), it can be done in randomized polynomial time. We will spell out the ideas later while discussing the algorithm part of Theorem 3.

### 4 MAIN RESULTS: HIGH DEGREE CIRCUITS

This section proves Theorems 1–2. 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 6.
4.1 Factors of a circuit with low-degree radical: Proof of Theorem 1

We simultaneously find the approximations of all the power series roots \( g_i \) of \( f(\tau \bar{\mathbb{X}}) \). At each recursive step, we get a better precision wrt degree. We show that knowing the approximations \( g_i^{< \delta} \), of \( g_i \) (for all \( i \)) up to degree \( \delta - 1 \), is enough to calculate approximations of \( g_i \) (simultaneously for all \( i \)) up to degree \( \delta \).

In this section, we use Theorem 17.

**Proof of Theorem 1.** From the hypothesis \( f = u_0 u_1 \). Define \( \deg(f) := d \). Suppose \( u_1 = h_1^1 \cdots h_m^m \), where \( h_i \)'s are coprime irreducible polynomials. Let \( d_0 \) be the degree of \( \text{rad}(u_1) = \prod_i h_i \). Note that \( \deg(h_i), m \leq d_0 \) and the multiplicity \( \epsilon_i \leq d \leq 2^i \), 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 \( \text{poly}(sd_0) \).

Using Theorem 17, we have \( \tilde{f}(\tau, y) = f(\tau \bar{\mathbb{X}}) = k \cdot u_0 \tau(\bar{\mathbb{X}}) \cdot \prod_{i \in [d_0]} (y - g_i)^{\tau_i} \), with \( g_i(\overline{0}) := \mu_i \) being distinct nonzero field elements. As \( \deg(h_i) \leq d_0 \); from Corollary 18, we deduce that for nonzero \( k_i \in \mathbb{F} \),

\[
 h_i(\tau \bar{\mathbb{X}}) \equiv k_i \cdot \prod_{i \in [d_0]} (y - g_i^{< d_i})^{\delta_i} \mod \mathcal{I}^{d_0+1} \quad \text{where} \quad I := (x_1, \ldots, x_n)
\]

where exponent \( \delta_i \in \{0, 1\} \). We can get \( h_i \) by applying \( \tau^{-1} \). Hence, it is enough to bound the size of \( g_i^{< d_i} \).

Let \( \bar{u}_0 := u_0(\tau \bar{\mathbb{X}}) \). From the repeated applications of Leibniz rule of the derivative (\( \partial_y(FG) = (\partial_y F)G + F(\partial_y G) \)), we deduce a classic logarithmic derivative identity,

\[
 \frac{\partial g}{\partial f} \equiv \frac{\partial g \bar{u}_0}{u_0} + \sum_{i=1}^{d_0} \frac{Y_i}{(y - g_i)}.
\]

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 \( \mathbb{F}\langle \bar{\mathbb{X}} \rangle \) when \( y \) is assigned any value \( \epsilon_i \in \mathbb{F} \), which is not equal to \( \mu_i \). We intend to find \( g_i \) mod \( \mathcal{I}^\delta \) inductively, for all \( \delta \geq 1 \). We assume that \( \mu_i \)'s and \( y_i \)'s are known. Suppose, we have recovered up to \( g_i \) mod \( \mathcal{I}^\delta \) and we want to recover \( g_i \) mod \( \mathcal{I}^\delta+1 \). The relevant recurrence, for \( \delta \geq 1 \), is:

**Claim 2 (Recurrence).**

\[
 \sum_{i=1}^{d_0} Y_i \cdot \frac{g_i^{< \delta}}{(y - \mu_i)^2} \equiv \frac{\partial g \bar{u}_0}{u_0} - \sum_{i=1}^{d_0} \frac{Y_i}{(y - g_i^{< \delta})} \mod \mathcal{I}^{\delta+1}.
\]

**Proof of Claim 2.** Using a power series calculation (Lemma 20), we have

\[
 \frac{1}{y - g_i} \equiv \frac{1}{y - (g_i^{< \delta} + g_i^{\geq \delta})} \equiv \frac{1}{y - g_i^{< \delta}} + \frac{g_i^{\geq \delta}}{(y - \mu_i)^2} \mod \mathcal{I}^{\delta+1}.
\]

Multiplying by \( y_i \) and summing over \( i \in [d_0] \), the claim follows. \( \square \)

We will compute \( g_i^{< \delta} \) incrementally; in particular knowing approximation up to the \( \delta - 1 \) homogeneous parts of \( g_i \), we will find the \( \delta \)-th part by solving a linear system. Concretely, assume that we have already computed a rational function \( g_i^{< \delta - 1} \) of the form \( g_i^{< \delta - 1} := C_{i, \delta - 1} / D_{i, \delta - 1} \) such that \( g_i^{< \delta - 1} \) approximates \( g_i \) correctly up to \( \delta - 1 \), i.e. \( g_i^{< \delta} \equiv g_i^{< \delta} \mod \mathcal{I}^\delta \).

In the free variable \( y \) in Claim 2, we plug-in \( d_0 \) random field values \( \epsilon_i \) and get the following system of linear equations:

\[
 M \cdot v_\delta = W_\delta, \quad \text{where} \quad M \in d_0 \times d_0 \quad \text{matrix while both} \quad v_\delta \quad \text{and} \quad W_\delta \quad \text{are both} \quad d_0 \times 1 \quad \text{matrices.} \quad \text{Both} \quad M \quad \text{and} \quad W_\delta \quad \text{are known matrices; in particular:
}
\]

\[
 M(i, j) := \frac{Y_j}{(\epsilon_i - \mu_j)^2}, \quad W_\delta(i) := \left( \frac{\partial g \bar{u}_0}{u_0} \right)_{y = \epsilon_i} - \bar{C}_{i, \delta} \quad \text{where} \quad \bar{C}_{i, \delta} := \sum_{j=1}^{d_0} \frac{Y_j}{(\epsilon_i - g_i^{< \delta - 1})^2}
\]
We want to solve for the unknown \(v_\delta\) whose \(i\)-th entry is \(v_\delta(i)\). We define the rational function \(g'_{i,\delta}\) iteratively as follows:

\[
g'_{i,\delta} = \begin{cases} 
\mu_i & \text{when } \delta = 0 \\
\frac{g'_{i,\delta-1} + v_\delta(i)}{\delta} & \text{otherwise}
\end{cases}
\]

We show that \(g'_{i,\delta}\) approximates \(g_i\) up to \(\delta\) where we crucially use the fact that \(M\) is invertible (Lemma 19).

**Claim 3 (Self-correction).** Let \(i \in [d_0]\) and \(\delta \geq 0\). Then, \(g'_{i,\delta} = g_i^{\leq \delta}\) mod \(I^{\delta+1}\).

**Proof of Claim 3.** We prove this by induction on \(\delta\). It is true for \(\delta = 0\) by definition. Suppose it is true for \(\delta - 1\). This means we have \(g'_{i,\delta-1} = g_i^{\leq \delta}\) mod \(I^{\delta}\) for all \(i\). Note that, \(g'_{i,\delta-1} \in \mathbb{F}(\mathbb{X}) \cap \mathbb{F}[[\mathbb{X}]]\). Let us write

\[
g'_{i,\delta-1} = g_i^{\leq \delta} + A_{i,\delta} + A'_{i,\delta}\quad \text{where } A'_{i,\delta} = 0 \text{ mod } I^{\delta+1}
\]

Here \(A_{i,\delta}\) is homogeneous of degree \(\delta\). Hence, for \(i \in [d_0]\), the linear constraint is:

\[
\sum_{j=1}^{d_0} Y_j \cdot \frac{v_\delta(j) - g'_{j,\delta-1}}{(c_i - \mu_j)^2} = \frac{\partial v_\delta}{\partial \mu_0} - \frac{\partial \mu_0}{\partial \mu_0} - \sum_j Y_j \cdot \frac{g_j^{\leq \delta}}{(c_i - g'_{j,\delta-1})} \mod I^{\delta+1}
\]

The “garbage” term \(A_{i,\delta}\) in RHS can be isolated using Lemma 20 as:

\[
\frac{1}{(c_i - g'_{j,\delta-1})} = \frac{1}{(c_i - (g_{j,\delta} + A_{i,\delta}))} = \frac{1}{(c_i - g_{j,\delta} - A_{i,\delta})} \mod I^{\delta+1}
\]

Plugging Eqn. (2) into (1), we get:

\[
\sum_{j=1}^{d_0} Y_j \cdot \frac{v_\delta(j) - g'_{j,\delta-1}}{(c_i - \mu_j)^2} = \frac{\partial v_\delta}{\partial \mu_0} - \frac{\partial \mu_0}{\partial \mu_0} - \sum_j Y_j \cdot \frac{g_j^{\leq \delta}}{(c_i - g'_{j,\delta-1})} = \sum_{j=1}^{d_0} Y_j \cdot \frac{A_{j,\delta}}{(c_i - \mu_j)^2} \mod I^{\delta+1}.
\]

Rewriting this, using Claim 2, we get:

\[
\sum_{j=1}^{d_0} Y_j \cdot \frac{v_\delta(j) + A_{i,\delta}}{(c_i - \mu_j)^2} \equiv \sum_{j=1}^{d_0} Y_j \cdot g_j^{\leq \delta} \mod I^{\delta+1}.
\]

Rearranging, we get that

\[
\sum_{j=1}^{d_0} Y_j \cdot \frac{v_\delta(j) + A_{i,\delta}}{(c_i - \mu_j)^2} \equiv 0 \mod I^{\delta+1}.
\]

As we vary \(i \in [d_0]\), we deduce, by Lemma 19, that \(v_\delta(j) + A_{i,\delta} - g_j^{\leq \delta} \equiv 0 \mod I^{\delta+1}\). Hence, for all \(j \in [d_0]\):

\[
g'_{i,\delta} = g'_{i,\delta-1} + v_\delta(j)
\]

\[
\equiv (g_j^{\leq \delta} + A_{j,\delta}) + (g_j^{\leq \delta} - A_{j,\delta})
\]

\[
\equiv g_j^{\leq \delta} \mod I^{\delta+1}.
\]

\[\square\]

**Lemma 19 (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 \(\frac{1}{(\mu_i - \mu_j)^2}\). Its entries are in the function field \(\mathbb{F}(\mathbb{X})\). Then, \(\det(A) \neq 0\).
Corollary. As det\(A\) \(\in \mathbb{F}(y) \setminus \{0\}\), using Polynomial Identity Lemma (Lemma 8), it follows that the matrix \(M\) where \(M(i, j) := \frac{1}{(y_i - y_j)^2}\) for \(c_i, c_j \in \mathbb{F}\), is invertible.

Proof of Lemma 19. The idea is to consider the power series of the function \(1/(y_i - y_j)^2\) and show that a monomial appears nontrivially in that of det\(A\).

We first need a claim about the coefficient operator on the determinant.

Claim 4. Let \(f_j = \sum_{i \geq 0} \beta_{i,j} x^i\) be a power series in \(\mathbb{F}[x]\), for \(j \in [d]\). Then, \(\text{Coeff}_{x^i} \circ \text{det}(f_j(x)) = \text{det}(\beta_{j,\alpha})\).

Proof of Claim 4. Observe that the rows of the matrix have disjoint variables. Thus, \(x_i^{\alpha_i}\) could be produced only from the \(i\)-th row. This proves:

\[
\text{Coeff}_{x^i} \circ \text{det}(f_j(x)) = \text{det} \left( \text{Coeff}_{x^i} \circ f_j(x) \right) = \text{det}(\beta_{j,\alpha}) .
\]

By Taylor expansion we have

\[
\frac{1}{(x - \mu)^2} = \frac{1}{\mu^2} \sum_{j \geq 1} j \left( \frac{x}{\mu} \right)^{j-1}.
\]

Hence, the coefficient of \(y_i^{j-1}\) in \(A(i, j)\) is

\[
\frac{1}{\mu_j^2} \cdot \frac{i}{\mu_j^{j-1}} = \frac{i}{\mu_j^{j+1}}.
\]

By the above claim, the coefficient of \(\prod_{i \in [d]} y_i^{\alpha_i - 1}\) in \(\text{det}(A)\) is: \(\det \left( \frac{1}{\mu_i} \right)\). By cancelling \(i\) (from each row) and \(1/\mu_j^2\) (from each column), we simplify it to the Vandermonde determinant:

\[
\det \begin{bmatrix}
\frac{1}{\mu_1} & \frac{1}{\mu_2} & \cdots & \frac{1}{\mu_d} \\
\frac{1}{\mu_1^2} & \frac{1}{\mu_2^2} & \cdots & \frac{1}{\mu_d^2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{1}{\mu_1^{d-1}} & \frac{1}{\mu_2^{d-1}} & \cdots & \frac{1}{\mu_d^{d-1}}
\end{bmatrix} = \prod_{i < j \in [d]} \left( \frac{1}{\mu_i} - \frac{1}{\mu_j} \right) \neq 0.
\]

Hence, the determinant of \(A\) is nonzero.

Remarks. (1) 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 \(\prod_{i \in [d]} y_i^{\alpha_i - 1}\) in \(\text{det}(A)\) for a set \(S = \{s_1, \ldots, s_d\}\) of distinct non-negative integers that are not divisible by \(p\). Moreover, one must consider ‘random’ \(\mu_i\)’s to deduce \(\text{det}(A) \neq 0\).

(2) The matrix \(A\) defined by \(A(i, j) := 1/(y_i - y_j)\) is the famous Cauchy matrix. It is not hard to show that the same proof of Lemma 19 works to establish the Cauchy matrix’s invertibility. For details on Cauchy matrix, see [Wik].

The following lemma is crucially used in Equation 2.

Lemma 20 (Series inverse). Let \(\delta \geq 1\). Assume that \(A\) is a polynomial of degree \(< \delta\) and \(B\) is a homogeneous polynomial of degree \(\delta \geq 1\), such that \(A(\overline{0}) \equiv \mu \neq 0\). Then, we have the following identity in \(\mathbb{F}[\overline{\langle x \rangle}]/(y) \cap \mathbb{F}[\overline{\langle x \rangle}][[y]]\):

\[
\frac{1}{y - (A + B)} \equiv \frac{1}{y - A} + \frac{B}{(y - \mu)^2} \mod \langle \overline{x} \rangle^{\delta+1}
\]
Remark. Since \( \mu \neq 0 \) and \( \delta > 1 \), the expression \( 1/(y-(A+B)) \) is a power series in both \( \mathbb{F}[\mathbb{Y}] \) and \( y \) (and also a rational function in \( y \), with coefficients from \( \mathbb{F}[\mathbb{Y}] \)). This can be easily seen just by considering \(( -1/(A+B) \cdot 1/(1-y/(A+B)) \) and noting the fact that \( 1/(A+B) \in \mathbb{F}[\mathbb{Y}] \).

Proof. We will use the notation \( A^{[1,\delta-1]} \) to refer to the sum of the homogeneous parts of \( A \) of degrees between 1 and \( \delta - 1 \) (equivalently, it is \( A^{-\delta} - \mu \)). Note that \( B \cdot A^{[1,\delta-1]} \) vanishes mod \( \langle \mathbb{Y} \rangle^{\delta+1} \). Now, in \( \mathbb{F}[\mathbb{Y}][\{y\}] \).

\[
\begin{align*}
\frac{1}{y-(A+B)} & \quad \frac{1}{y-\mu - (A^{[1,\delta-1]}+B)} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 - A^{[1,\delta-1]}+B} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 + A^{[1,\delta-1]}+B} + \frac{(A^{[1,\delta-1]}+B)^2}{y-\mu} + \ldots \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 + A^{[1,\delta-1]}+B} + \frac{(A^{[1,\delta-1]}+B)^2}{y-\mu} + \frac{(A^{[1,\delta-1]}+B)^3}{y-\mu} + \ldots \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 + A^{[1,\delta-1]}+B} + \frac{B}{(y-\mu)^2} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 - A^{[1,\delta-1]}+B} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-\mu} & \quad \frac{1}{1 - A^{[1,\delta-1]}+B} - \frac{B}{(y-\mu)^2} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\frac{1}{y-A} & \quad \frac{1}{y-\mu} \mod \langle \mathbb{Y} \rangle^{\delta+1} \\
\end{align*}
\]

□

Size analysis: Here we give the overall process of finding factors using the 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 \( g_i^{\leq 5} \) or \( g_i' \).

At the \(( \delta - 1)\)-th step of the 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 \( \mathbb{F}[\mathbb{Y}] \). 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 6). 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 there exist field elements \( \beta_{ij} \) such that

\[
v_\delta(i) = \sum_{j=1}^{d_i} \beta_{ij} \cdot W_\delta(j) = \sum_{j=1}^{d_i} \beta_{ij} \left( \left[ \frac{\partial y}{\partial f} - \frac{\partial g_0}{\partial u_0} \right]_{y=c_j} \right)
\]

Initially, we precompute, for all \( j \in \{d_0\} \), \( \langle (\partial y/f - \partial g_0/u_0) \rangle_{y=c_j} \). Note that \( \partial y/f \) has poly(s) size circuit (high degree of the circuit does not matter, see Lemma 7). Invertibility of \( f_{y=c_j} \) and \( u_0_{y=c_j} \) follows from the fact that we chose \( c_j \)'s randomly. In particular, \( f(0, y) \), and so \( u_0(0, y) \), have roots in \( \mathbb{F} \) which are distinct from \( c_j, j \in \{d_0\} \). Thus, \( f(\mathbb{Y}, c_j) \) and \( u_0(\mathbb{Y}, c_j) \) have nonzero constants and so are invertible in \( \mathbb{F}[\mathbb{Y}] \). Similarly, \( y_1/(c_j - g_{i,'\delta-1}/c_j) \) exists in \( \mathbb{F}[\mathbb{Y}] \).
Thus, the matrix recurrence allows us to calculate the polynomials $C_{i,\delta}$ and $D_{i,\delta}$, given their $\delta - 1$ analogs, by adding $\text{poly}(d_0)$ many wires and nodes. The precomputations cost us size $\text{poly}(s, \delta)$. Hence, both $C_{i,\delta}$ and $D_{i,\delta}$ has $\text{poly}(s, \delta, d_0)$ sized circuit.

We can assume we have only one division gate at the top, as for each gate $G$ we can keep track of the numerator and the denominator of the rational function computed at $G$, and easily simulate all the algebraic operations in this representation. When we reach precision $\delta = d_0$, we can eliminate the division gate at the top. As $D_{1,d_0}$ is a unit, we can compute its inverse using the power series inverse formula and approximate only up to degree $d_0$ (Lemma 5). Finally, the circuit for the polynomial $g_i^{d_0} \equiv C_{i,d_0}/D_{i,d_0} \mod \ell^{d_0+1}$, for all $i \in [d_0]$, has size $\text{poly}(s, d_0)$.

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

4.2 Low degree factors of general circuits: Proof of Theorem 2

As a consequence of the reduction presented in Section 3.1, a direct approach to proving the Factor Conjecture is via a direct approach to proving the Factor Conjecture.

4.2 Low degree factors of general circuits: Proof of Theorem 2

As a consequence of the reduction presented in Section 3.1, a direct approach to proving the Factor Conjecture is via computing arithmetic circuits of small size giving approximations (up to some low degree) of power series roots that have high multiplicity. First, we need to find methods for approximating roots with multiplicity $\geq 2$. The classical Newton iteration formula fails here, but a simple modification of Newton iteration works if we know the multiplicity of the root. In the numerical analysis literature [DB08], this is known as modified/generalized Newton iteration with multiplicity. Using this method, we give a plausible approach to prove the Factor Conjecture.

Newton Iteration with multiplicity: The following is a generalized Newton Iteration formula, as it works with any multiplicity $e > 0$.

**Lemma 21 (NI with multiplicity).** If $f(x, y) = \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i}$, where $g_i \pmod 1$ are non-zero and distinct, and $\gamma_i > 0$, then the each power series $g_i$ can be approximated by the recurrence:

$$y_{t+1} := y_t - \gamma_i \cdot \frac{f}{\partial_y f} \bigg|_{y=y_t}$$

(3)

where $y_t \equiv g_i \pmod {\ell^{2t}}$.

**Proof.** We will show the above for $g_1$. We remark that $y_t$ is a rational function; this is easy to prove by simple induction (on $t$). Rewrite

$$\frac{\partial_y f}{f} = \sum_{i=1}^{d_0} \frac{\gamma_i}{(y - g_i)} = \frac{(1 + L_1) \cdot \gamma_1}{(y - g_1)}$$

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 $y = y_t := g_1^{2^{2t}}$, then $y_t - g_1 = g_1^{2^{2t}} - g_1$ is a unit in $\mathbb{F}[\mathbb{F}[[x]]]$ for $i \neq 1$ (because it is a nonzero constant mod $\ell$). Hence, $y_t - g_1 = g_1^{2^{2t}} - g_1 \equiv 0 \pmod {\ell^{2t}}$. Together they imply that $L_1|_{y=y_t} \equiv 0 \pmod {\ell^{2t}}$. Hence,

$$f/\partial_y f|_{y=y_t} \equiv (y_t - g_1)/\gamma_1 \cdot (1 + L_1)^{-1} \equiv ((y_t - g_1)/\gamma_1) \cdot (1 - L_1 + L_1^2 - \ldots) \equiv (y_t - g_1)/\gamma_1 \pmod {\ell^{2t+1}}.$$  (4)

The last expression implies that $y_t - y_1 \cdot f/\partial_y f|_{y=y_t} \equiv g_1 \pmod {\ell^{2t+1}}$, as desired. □

**Remarks.**

(1) Note that, when $\gamma_1 = 1$, (i.e. $g_1$ is a simple root of $f$), the above is an alternate proof of the classical Newton Iteration (NI) [New69] that finds a simple root in a recursive way (see Lemma 15).

(2) There is a subtle point about Equation 3 when $\gamma_1 \geq 2$. The denominator $\partial_y f|_{y=y_t}$ is zero mod $\ell$, thus, its reciprocal does not exist! However, the ratio $(f/\partial_y f)|_{y=y_t}$ does exist in $\mathbb{F}[[x]]$ (Equation 4), one can think of first reducing this...
Discovering the roots: Uniform closure results for algebraic classes under factoring

5 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 fields like \( \mathbb{Q}, \mathbb{Q}_p \), or \( \mathbb{F}_q \) for prime-power \( q \), we also give efficient randomized algorithms to output the complete factorization of polynomials belonging to that class (stated as Theorem 26). We use the notation \( g \mid f \) to denote that \( g \) divides \( f \) but \( g^d \) does not divide \( f \). Again, we denote \( l := \langle x_1, \ldots, x_n \rangle \).

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 algorithmic approach.
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 17 we have that \( \tilde{f}(\bar{x}, y) := f(\bar{x} + \gamma) \) splits like \( f = \prod_{i \geq 1} (y - g_i)^{\lambda_i} \), with \( g_i(0) = \mu_i \) 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 a formula of size \( n^{O(\log n)} \). To show size bound for all the factors, it is enough to show that the approximations of the power series roots, i.e. \( g_i^{\tilde{d}} \) has size \( n^{O(\log n)} \) size formula. This follows from the reduction of factoring to approximations of power series roots (Section 3.1).

We differentiate \( \tilde{f} \) wrt \( y \), \( (y_i - 1) \) many times, so that the multiplicity of the root we want to recover becomes exactly one. The differentiation would keep the size \( \text{poly}(n^{\log n}) \) (Lemma 7). Now, we have \( (y - g_i) || f^{(y-1)} \), and we can apply classical Newton iteration formula (Lemma 15). For all \( 0 \leq t \leq \log d + 1 \), we compute \( A_t \) and \( B_t \) such that \( A_t/B_t \equiv g_i \mod I^d \). Moreover, \( B_t \) is invertible in \( \mathbb{F}[\mathbb{T}] \) (because \( g_i \) is a simple root of \( f^{(y-1)} \)).

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 \( s \) (which is \( \text{size}(f) \)) many copies at each step. We claim that it can be reduced to only \( d^2 \) many copies.

We can pre-compute (with blow-up at most \( \text{poly}(sd) \)) all the coefficients \( C_0, \ldots, C_d \) wrt \( y \), given the formula of \( f := C_0 + C_1 y + \cdots + C_d y^d \) using interpolation. We can do the same for the derivative formula. For details on this interpolation trick, see Section 2.1. Using interpolation, we can convert the formula of \( \tilde{f} \) and its derivative to the form \( C_0 + C_1 y + \cdots + C_d y^d \). 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 \( g_i^{<d^2} \) to get \( g_i^{<d^2+1} \). This leads to \( d^2 \) blow up at each step of the iteration.

As the denominators \( B_t \) 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 \( \text{poly}(sd) \) (Lemma 6).

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

Hence, \( g_i^{<d} \equiv A_{\log d + 1}/B_{\log d + 1} \mod I^{d+1} \) has \( n^{O(\log n)} \) size formula, and so does every polynomial factor of \( f \) after applying \( \tau^{-1} \).

Size analysis for ABP: This analysis is similar to that of the formula model, as the size blow-up in each NI iteration for differentiation, division, and truncation (to degree \( \leq d \)) is the same as that for formulas. A noteworthy difference is that we need to eliminate division in every iteration (Lemma 5) and we cannot postpone it. This leads to a blow-up of \( d^4 \) in each step. Hence, \( S_{\log d + 1} = \text{poly}(sd) \cdot d^4 \log d = \text{poly}(n^{\log n}) \).

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 parameter. Now, our given polynomial \( f \) has \( n^{O(\log n)} \) size parameters. As before, it is easy to show that \( g_i^{<d} \) has \( n^{O(\log n)} \) size parameters.

For the pre-processing (taking \( y_1 - 1 \)-th derivative of \( \tilde{f} \) wrt \( y \)), the blow-up in the size parameters is only \( \text{poly}(n^{\log n}) \).

Now we analyze the blow-up due to classical Newton iteration. We compute \( A_t \) and \( B_t \) such that \( A_t/B_t \equiv g_i \mod I^d \).

Using the closure properties of VNP (discussed in Section 2.1), 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 \( \max(\text{size}(A_t), \text{size}(B_t)) \leq S_t \) and witness size \( \leq W_t \). Then we have \( S_{t+1} \leq O(d^4 S_t) + \text{poly}(n^{\log n}) \). So, finally, we have \( S_{\log d + 1} = \text{poly}(sd) \cdot d^4 \log d = \text{poly}(n^{\log n}) \). It is clear that \( g_i^{<d} \equiv A_{\log d + 1}/B_{\log d + 1} \mod I^{d+1} \).
1 has poly($n^{\log n}$) size verifier circuit. The same analysis works for $W_f$, and the witness size remains $n^{O(\log n)}$.

Moreover, we get the corresponding bounds for every polynomial factor of $f$ after applying $\tau^{-1}$.

Remark. Recently, Chou, Kumar and Solomon [CKS19b] have improved our result on VNP, showing that VNP is closed under factors. Also note that, we get a short non-algorithmic proof of the closure of VP under factors using Newton iteration and the reduction of factor computation to approximating power series roots (Section 3.1). [CKS19a] gave another short proof for the same using the multidimensional version of Newton iteration.

5.1 Randomized factoring algorithm for formulas and ABPs

This subsection is dedicated to the design and analysis of the constructive (algorithmic) part to factorize a given poly($n$) degree and poly($n^{\log n}$) size formula or ABP.

We need the following lemma (adapted from [KSS15]) that discusses how to perform linear algebra when the coefficients of vectors are given as formulas (respectively ABPs).

**Lemma 22.** (Linear algebra using PIT [KSS15, Lemma 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 $\mathbb{F}[\mathbf{x}]$. Suppose, we have an algebraic formula (respectively ABP) of size $\leq n^{O(\log n)}$ computing each entry. Then, there is a randomized poly($n^{\log n}$)-time algorithm that either:

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

**Proof.** This was proved in [KSS15, Lemma 2.6] for the circuit model. Since we are using a different model, we repeat the details. The idea is the following. Iteratively, for every $r = 1, \ldots, n$ we shall find an $r \times r$ minor contained in the first $r$ columns that is full rank. While continuing this process, we either reach $r = n$ in which case it means that the matrix has full column rank, hence, $u = 0$ is the only solution, or we get stuck at some value say $r = r_0$. We use the fact that $r_0$ is rank, and using this minor we construct the required nonzero vector $u$.

We explain the process in a bit more detail. Using a randomized algorithm, we look for some nonzero entry in the first column. If no such entry is found we can simply take $u = (1, 0, \ldots, 0)$. So assume that such a nonzero entry is found. After permuting the rows we can assume wlog that this is $M_{1,1}$. Thus, we have found a $1 \times 1$ minor satisfying the requirements. Assume that we have found an $r \times r$ full rank minor that is composed of the first $r$ rows and columns (we can always rearrange and hence it can be assumed wlog that they correspond to first $r$ rows and columns). Denote this minor by $M_r$.

Now for every $(r + 1) \times (r + 1)$ submatrix of $M$ contained in the first $r + 1$ columns and containing $M_r$, we check whether the determinant is 0 by randomized algorithm (Lemma 8). If any of these submatrices have nonzero determinant, then we pick one of them and call it $M_{r+1}$. Otherwise, we have found that first $r + 1$ columns of $M$ are linearly dependent.

As $M_r$ is full rank, there is $v \in \mathbb{F}(\mathbf{x})^r$ such that $M_r v = (M_{1,r+1}, \ldots, M_{r,r+1})^T$. This can be solved by applying Cramer’s rule. The $i$-th entry of $v$ is of the form $\det(M_{r}^{(i)})/\det(M_r)$, where $M_r^{(i)}$ is obtained by replacing $i$-th column of $M_r$ with $(M_{1,r+1}, \ldots, M_{r,r+1})^T$. Observe that $\det(M_r)$, as well as $\det(M_r^{(i)})$, are both in $\mathbb{F}[\mathbf{x}]$.

Then it is immediate that $u := (\det(M_r^{(1)}), \ldots, \det(M_r^{(r)}), -\det(M_r), 0, \ldots, 0)^T$ is the desired vector.

To find $M_r$, each time we have to calculate the determinant and decide whether it is 0 or not. This is simply PIT for a determinant polynomial with entries of algebraic complexity $n^{O(\log n)}$ and degree $n^{O(1)}$. So, we have a comparable randomized algorithm for this. Determinant of a symbolic $n \times n$ matrix has $n^{O(\log n)}$ size formula (respectively poly($n$) ABP) [MV97]. When the entries of the matrix have $n^{O(\log n)}$ size formula (respectively ABP), altogether, the determinant...
polynomial has the same algebraic complexity. There are \(< n^2 \) PIT invocations to test zeroeness of the determinant. Altogether, we have a \( \text{poly}(n^{\log n}) \)-time randomized algorithm for this (Lemma 8).

Before moving to the constructive part, we discuss a new method for computing \( \text{gcd} \) of two polynomials, which not only fits well in the algorithm but is also of independent interest. We recall the definition of \( \text{gcd} \) of two polynomials \( f, g \) in the ring \( \mathbb{F}[\mathbb{X}] \): \( \text{gcd}(f, g) := h \iff h|f, h|g \) and \((h'|f, h'|g \Rightarrow h'|h) \). It is unique up to constant multiples.

**Claim 5** (Computing formula \( \text{gcd} \)). Given two polynomials \( f, g \in \mathbb{F}[\mathbb{X}] \) of degree \( d \) and computed by a formula (respectively \( \text{ABP} \)) of size \( s \). One can compute a formula (respectively \( \text{ABP} \)) for \( \text{gcd}(f, g) \), of size \( \text{poly}(s, d^{\log d}) \), in randomized \( \text{poly}(s, d^{\log d}) \) time.

**Proof of Claim 5.** The idea is the following. Suppose, \( \text{gcd}(f, g) = h \) is of degree \( d > 0 \), then we will compute \( h(\mathbb{r} \mathbb{X}) \) for a random map \( \mathbb{r} \) as in Theorem 17. We know wlog that \( \mathbb{f} := f(\mathbb{r} \mathbb{X}) = \prod_i (y - A_i)^{a_i} \) and \( \mathbb{g} := g(\mathbb{r} \mathbb{X}) = \prod_j (y - B_j)^{b_j} \), where \( A_i, B_j \in \mathbb{F}[\mathbb{X}] \). Since \( \mathbb{F}[\mathbb{X}] \subset \mathbb{F}[\mathbb{X}] \) are UFDs (Proposition 16), we could say wlog that \( h(\mathbb{r} \mathbb{X}) = \prod_{i \in S} (y - A_i)^{\min(a_i, b_i)} \), where \( S = \{ i \mid A_i = B_i \} \) after possible rearrangement. Now, as \( \mathbb{r} \) is a random invertible map, we can assume that, for \( i \neq j, A_i \neq B_j \) and that \( A_i(\mathbb{0}) \neq B_j(\mathbb{0}) \) (Lemma 13). So, it is enough to compute \( A_i^{\varepsilon d} \) and \( B_i^{\varepsilon d} \) and compare them using evaluation at \( \mathbb{0} \). If indeed \( A_i = B_i \), then \( A_i^{\varepsilon d} = B_i^{\varepsilon d} \). 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 \( b_i \), additionally, we get the corresponding starting points of classical Newton iteration, i.e. \( A_i(\mathbb{0}) \) and \( B_i(\mathbb{0}) \)’s. Using \( \text{NI} \), one can compute \( A_i^{\varepsilon d} \) and \( B_i^{\varepsilon d} \), for all \( i \). Suppose, after rearrangement of \( A_i \) and \( B_i \)’s (if necessary), we have \( A_i = B_i \) for \( i \in [s] \) \( = S \) and \( A_i \neq B_j \) for \( i \in [s + 1, d] \). Lemma 13 can be used to deduce that \( A_i(\mathbb{0}) \neq B_j(\mathbb{0}) \) for \( i, j \in [1, d] - S \). So, we have in \( \text{gcd}(\mathbb{f}, \mathbb{g}) = \prod_{i \in S} (y - A_i)^{\min(a_i, b_i)} \); the index set \( S \), the exponents and \( A_i(\mathbb{0}) \)’s computed.

**Size analysis:** To compute \( A_i^{\varepsilon d} \) (similarly \( B_i^{\varepsilon d} \)), we use the classical newton iteration (Lemma 15) after differentiating (up to order to make multiplicity-1) to make \( A_i \) a simple root (i.e. multiplicity 1) of the differentiated polynomial. That leads to a polynomial blowup in size (Lemma 7). It is clear that at each \( \text{NI} \) step, there will be a multiplicative \( d^2 \) blow up (due to interpolation, division and truncation). There are \( \log d \) iterations in \( \text{NI} \). Altogether the truncated roots have \( \text{poly}(s, d^{\log d}) \) size formula (respectively \( \text{ABP} \)). This directly implies that \( \text{gcd}(\mathbb{f}, \mathbb{g}) \) has \( \text{poly}(s, d^{\log d}) \) size formula (respectively \( \text{ABP} \)). By taking the product of the linear factors, truncating to degree \( d \), and applying \( \tau^{-1} \), we can compute the polynomial \( \text{gcd}(f, g) \).

Randomization is needed for \( \tau \) 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 to \( \mathbb{F} \). For details, see Section 6.2.

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

1. Choose \( \mathbb{n}, \mathbb{b} \in \mathbb{F}^m \) and apply \( \tau : x_i \rightarrow x_i + \alpha_i y + \beta_i \). Denote the transformed polynomial \( f (\mathbb{r} \mathbb{X}) \) by \( \mathbb{f} (\mathbb{r} \mathbb{X}, y) \). Wlog, from Theorem 17, \( \mathbb{f} \) has factorization of the form \( \prod_{i=1}^{\mu_i} (y - g_i)^{\nu_i} \), where \( \mu_i := g_i(\mathbb{0}) \) are distinct.

2. Factorize \( \mathbb{f}(\mathbb{0}, y) \) over \( \mathbb{F}[y] \). This will give \( \mu_i \) and \( \nu_i \)’s.

3. Fix \( i = \mu_0 \). Differentiate \( \mathbb{f} \), wrt \( y \), \( (\nu_0 - 1) \) many times to make \( g_{\mu_0} \) 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 \( \mu_0 \mod (d) \). We get \( g_{\mu_0}^{\varepsilon 2^k} \) at the end of the process (mod \( I^{2^k} \)).
(5) Apply the transformation $x_i \mapsto T x_i$ ($T$ acts as a degree-counter). Consider $\tilde{g}_{ih} \coloneqq g_{ih}^{\text{cd}}(T \bar{x})$. Solve the following homogeneous linear system of equations, over $\mathbb{F}[\bar{x}]$, in the unknowns $u_{ij}$ and $v_{ij}$'s,

$$\sum_{0 \leq i+j < d} u_{ij} \cdot y^i T^j = (y - \tilde{g}_{ih}) \cdot \sum_{0 \leq i+j < d} v_{ij} \cdot y^i T^j \mod T^{2^k}.$$ 

Solve this system, using Lemma 22, to get a nonzero polynomial (if one exists) $u \coloneqq \sum_{0 \leq i+j < d} u_{ij} \cdot y^i T^j$.

(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 $G(\bar{x}, y, T) \coloneqq \gcd_y(u(\bar{x}, y, T), \hat{f}(T \bar{x}, y))$ using Claim 5.

(9) Compute $G(\bar{x}, y, 1)$ and transform it by $r^{-1}: x_i \mapsto x_i - \alpha_i y - \beta_i$, $i \in [n]$, and $y \mapsto y$. Output this as an irreducible polynomial factor of $f$.

Claim 6 (Existence). If $f$ is reducible, then the linear system (Step 5) has a non-trivial solution.

Proof of Claim 6. If $f$ is reducible, then let $f = \prod f_i^{\alpha_i}$ be its prime factorization. Assume wlog that $y - g_{ih} | \hat{f}_1 \coloneqq f_1(T \bar{x})$.

Of course $0 < \deg_y(\hat{f}_1) = \deg(f_1) < d$.

Observe that we are done by picking $u$ to be $\hat{f}_1(T \bar{x}, y)$. For, total degree of $f_1$ is $< d$, and so that of $\hat{f}_1(T \bar{x}, y)$ wrt the variables $y, T$ is $< d$.

Moreover, $y - g_{ih} | \hat{f}_1 \implies \hat{f}_1 \equiv (y - g_{ih}) v, \text{ for some } v \in \mathbb{F}[\bar{x}][y]$ with $\deg_y < d$. Hence,

$$\hat{f}_1 \equiv (y - g_{ih}) \cdot v \mod T^{2^k} \implies u \equiv (y - g_{ih}) \cdot v(T \bar{x}, y) \mod T^{2^k}$$

This shows the existence of a nontrivial solution of the linear system (Step 5).

Claim 7 (Step 8’s success). If the linear system (Step 5) has a non-trivial solution, then $0 < \deg_y G \leq \deg_y u < d$.

Proof of Claim 7. Suppose $(u, v)$ is the solution provided by the algorithm in Lemma 22 ($u$ being in the unknown LHS and $v$ being the unknown RHS). Consider $G = \gcd_y(u, \hat{f}(T x, y))$. We know that there are polynomials $a$ and $b$ such that $au + b\hat{f}(T x, y) = \Res_y(u, \hat{f}(T x, y))$ (Section 2.2). Consider $\deg_T(\Res_y(u, \hat{f}(T x, y))$. As the degree of $T$ in $u$ and $\hat{f}(T x, y)$ can be at most $d$, hence degree of $T$ in resultant can be atmost $2d^2$ (Section 2.2). Clearly, $\deg_y G \leq \deg_y u < d$.

If $\deg_y G = 0$ then the resultant of $u, \hat{f}(T \bar{x}, y)$ wrt $y$ will be nonzero (Proposition 11). Suppose the latter happens.

Now, we have $u = (y - g_{ih}) v \mod T^{2^k}$. Since $y - g_{ih} \mid \hat{f}$ we get that $y - g_{ih}(T \bar{x}) \mid \hat{f}(T \bar{x}, y)$. Assume that $\hat{f}(T x, y) = (y - g_{ih}(T \bar{x})) \cdot w$.

Thus, we can rewrite the previous equation as: $au + b\hat{w}(T \bar{x}, y) \equiv (y - g_{ih})(aw + bw) \mod T^{2^k}$. Note that the latter is nonzero mod $T^{2^k}$ because the resultant is a nonzero polynomial of degree $T < 2^k$ putting $y = g_{ih}$ the LHS vanishes, but RHS does not (because it is independent of $y$). This gives a contradiction.

Thus, $\Res_y(u, \hat{f}(T x, y) = 0$. This implies that $0 < \deg_y G < d$.

Next we show that if one takes the minimal solution $u$ (wrt degree of $y$), it will correspond to an irreducible factor of $f$. We will use the same notation as above.
Claim 8 (Irred. factor). Suppose \( y - g_{i_k} | \tilde{f}_1 \) 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.

Proof of Claim 8. Suppose \( f \) is reducible, hence, as shown above, \( G \) is a non-trivial factor of \( \tilde{f}(T\bar{x}, y) \). Recall that \( \tilde{f}(T\bar{x}, y) = \prod_{i_1 \leq i \leq d_1} (y - g_i(T\bar{x}))^{y_j} \) is a factorization over \( \mathbb{F}[[T, x]] \). We have that \( y - g_{i_k} | G \mod T^{2k} \). Thus, \( y - g_{i_k} | G \) absolutely (because the power series ring is a UFD and use Theorem 17). So, \( y - g_{i_k} | G \) over the power series ring. Since, \( \tilde{f}_1(T\bar{x}, y) \) is an irreducible polynomial, we can deduce that \( \tilde{f}_1(T\bar{x}, y) | G \) in the polynomial ring. So, \( \deg_y(f_1) \leq \deg_y(G) \).

We have \( \deg_y(\tilde{f}_1(T\bar{x}, y)) = \deg(f_1) = d_1 \). By the above discussion, the linear system in the Step 7 will not have a solution of \( \deg_y(u) \) below \( d_1 \). Let us consider the linear system in the Step 7 that wants to find \( u \) of \( \deg_y(u) = d_1 \).

This system has a solution, namely the one with \( u := \tilde{f}_1(T\bar{x}, y) \mod T^{2k} \). Then, by the above claim, we will get the \( G \) as well in the subsequent Step 8. This gives \( \deg_y(G) \leq \deg_y(u) = d_1 \). With the previous inequality, we get \( \deg_y(G) = \deg_y(u) = \deg_y(f_1) \). In particular, \( G \) and \( \tilde{f}_1(T\bar{x}, y) \) are the same up to a nonzero constant multiple. □

Alternative to Claim 5: The above proof (Claim 8) suggests that the gcd question of Step 8 is rather special: One can just write \( u = \sum_{0 \leq i \leq d_1} g_i(T\bar{x}, x)^y \) and then compute the polynomial \( G = \sum_{0 \leq i \leq d_1} (c_i/c_{d_1}) \cdot y^i \) as a formula (respectively ABP), by eliminating division (Lemma 5).

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

The running time analysis of the algorithm is by now routine. If we start with an \( f \) computed by a formula (respectively ABP) of size \( n^{O(\log n)} \), then as observed before, one can compute \( g_{i_k} \) which has \( n^{O(\log n)} \) size formula (respectively ABP). This takes care of s 1-4.

Now, solve the linear system in s 5-7 of the algorithm. Each entry of the matrix is a formula (respectively ABP) size \( n^{O(\log n)} \). The time complexity is similar by invoking Lemma 22.

Step 8 is to compute gcd of two \( n^{O(\log n)} \) size formulas (respectively ABPs) which again can be done in \( n^{O(\log n)} \) time giving a size \( n^{O(\log n)} \) formula (respectively ABP) as discussed above.

This completes the randomized \( \text{poly}(n\log n) \)-time algorithm that outputs \( n^{O(\log n)} \) sized factors. □

Remarks. (1) The above results hold true for the classes \( VBP(s), VF(s), VNP(s) \) for any size function \( s = n^{O(\log n)} \).

(2) By using a reversal technique [Oli16, Section 1.1.2] and a modified \( \tau \), our size bound can be shown to be \( \text{poly}(s, d^{O(\log s)}) \), where \( r \) (respectively \( d \)) is the individual-degree (respectively degree) bound of \( f \). So, when \( r \) is constant, we get a factor as a poly(s)-size formula (respectively ABP). Oliveira [Oli16] proved the same result for formulas.

6 EXTENSIONS

6.1 Closure of approximative complexity classes

This section shows that all our closure under factoring results can be naturally generalized to corresponding approximative algebraic complexity classes.

In computer science, the notion of approximative algebraic complexity emerged in early works on matrix multiplication (the notion of border rank, see [BCS13]). It is also an important concept in the geometric complexity theory program (see [GMQ16]). The notion of approximative complexity can be motivated through two ways, topological and algebraic and both the perspectives are known to be equivalent. Both allow us to talk about the convergence \( \epsilon \to 0 \).
In what follows, we can see \( \epsilon \) as a formal variable and \( \mathbb{F}(\epsilon) \) as the function field. For an algebraic complexity class \( C \), the approximation is defined as follows [BIZ18, Defn.2.1].

**Definition 23 (Approximative closure of a class [BIZ18]).** Let \( C \) be an algebraic complexity class over field \( \mathbb{F} \). A family \( (f_n) \) of polynomials from \( \mathbb{F}[\mathcal{V}] \) is in the class \( \mathcal{C}(\mathbb{F}) \) if there are polynomials \( f_{n;i} \) and a function \( t : \mathbb{N} \rightarrow \mathbb{N} \) such that \( g_n \) is in the class \( C \) over the field \( \mathbb{F}(\epsilon) \) with \( g_n(\mathcal{V}) = f_n(\mathcal{V}) + \epsilon f_{n;1}(\mathcal{V}) + \epsilon^2 f_{n;2}(\mathcal{V}) + \cdots + \epsilon^t(n) f_{n;t}(\mathcal{V}) \).

The above definition can be used to define closures of classes like \( \mathcal{V} \), \( \mathcal{VBP} \), \( \mathcal{VP} \), \( \mathcal{VNP} \) which are denoted as \( \mathcal{V} \), \( \mathcal{VBP} \), \( \mathcal{VP} \), \( \mathcal{VNP} \) respectively. In these cases one can always ensure that the degrees of \( g_n \) and \( f_{n;i} \) are polynomial.

Following Bürgisser [Bür04] - Let \( K : = \mathbb{F}(\epsilon) \) be the rational function field in variable \( \epsilon \) 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 24.** [Bür04, Defn.3.1] Let \( f \in \mathbb{F}[x_1, \ldots, x_n] \). The approximative complexity \( \text{size}(f) \) 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 \( \leq r \).

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

Using this new notion of size, one can define the analogous class \( \mathcal{V} \). It is known to be closed under factors [Bür04, Theorem 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 \( \mathcal{V}, \mathcal{VBP}, \mathcal{VP} \) and \( \mathcal{VNP} \) the polynomials have poly(\( n \)) degree. So, by using repeated differentiation, we can assume the power series root (of \( \tilde{f} : = f(\tau \mathcal{V}) \)) to be simple (i.e. multiplicity=1) and apply classical NI. Here we give a brief sketch of the overall idea.

**Root finding using NI over \( K \).** For degree-\( d \) \( f \in \mathbb{F}[\mathcal{V}] \) if \( \text{size}(f) = s \) then: \( \exists F \in R[\mathcal{V}] \) with a size \( s \) circuit satisfying \( F|_{\epsilon = 0} = f \). The degree of \( F \) wrt \( \mathcal{V} \) may be greater than \( d \). In that case, we can extract the part up to degree \( d \) and truncate the rest [Bür04, Prop.3.1]. So log \( \deg_{\mathcal{V}}(F) = \deg(f) \).

By applying a random \( \tau \) (using constants \( \mathbb{F} \) we can assume that \( \tilde{F} := F(\tau \mathcal{V}) \in R[\mathcal{V}, y] \) is monic (i.e. leading-coefficient, wrt \( y \) in \( \tilde{F} \), is invertible in \( R \)). Otherwise, \( \deg_y(\tilde{F}) = \deg_y(\tilde{f}) = \deg_{\mathcal{V}}(f) \) 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 \( \tilde{F}|_{\mathcal{V} = 0} = \tilde{f} \) and both have their leading-coefficients 1 wrt \( y \).

Let \( \mu \) be a root of \( \tilde{f}(\bar{0}, y) \) of multiplicity one (as discussed before). Since \( \tilde{F}(\bar{0}, y) \equiv \tilde{f}(\bar{0}, y) \mod \epsilon \), we can build a power series root \( \mu(\epsilon) \in \mathbb{F}[\epsilon] \) of \( \tilde{F}(\bar{0}, y) \) using NI, with \( \mu \) as the starting point. But \( \mu(\epsilon) \) may not converge in \( K \). To overcome this obstruction, [Bür04] devised a clever trick.

Define \( \tilde{F} : = \tilde{F}(\mathcal{X}, y + \mu + \epsilon) - \tilde{F}(\bar{0}, \mu + \epsilon) \). Note that \( \tilde{F}(\bar{0}, 0) \) is a simple root of \( \tilde{F}(\mathcal{X}, y) \) [Bür04, Eqn.5]. So, a power series root \( y_{\text{new}} \) of \( \tilde{F} \) can be built iteratively by classic NI (Lemma 15):

\[
y_{t+1} := y_t - \frac{\partial \tilde{F}(\bar{0}, \mu + \epsilon)}{\partial y} |_{y = y_t} .
\]

Where, \( y_{\text{new}} \equiv y_t \mod (\bar{0})^d \). One can easily prove that \( y_t \) is defined over the coefficient field \( K \), using induction on \( t \).

Note that \( \tilde{F}|_{\epsilon = 0} = \tilde{f}(\mathcal{X}, y + \mu) - \tilde{f}(\bar{0}, \mu) = \tilde{f}(\mathcal{X}, y + \mu) \). So, \( y_{\text{new}} \) is associated with a root of \( \tilde{f} \) as well. This implies that by using several such roots \( y_{\text{new}} \), we can get an appropriate product \( \tilde{G} \in R[\mathcal{X}, y] \), such that an actual polynomial factor of \( f \) (over field \( \mathbb{F} \)) equals \( \tilde{G}|_{\epsilon = 0} \).

The above process, when combined with the first part of the proof of Theorem 3, does imply:
Theorem 25 (Approximative factors). The approximative complexity classes $\overline{VP}(n^{\log n})$, $\overline{VBP}(n^{\log n})$ and $\overline{VNP}(n^{\log n})$ are closed under factors.

The same question for the classes $\overline{VP}$, $\overline{VBP}$ and $\overline{VNP}$ we leave as an open question. (Though, for the respective bounded individual-degree polynomials we have the result as before.)

6.2 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 standard technique used in all the proofs is the structural result (Theorem 17) which talks about power series roots with respect to $y$. Recall that we use a random linear map $\tau : x_1 \mapsto x_1 + ay + \beta_i$, where $a, \beta_i \in \mathbb{F}$, to make the input polynomial $f$ monic in $y$ and the individual degree of $y$ equal to $d := \deg(f)$. If we set all the variables to zero except $y$, we get a univariate polynomial $\tilde{f}(0, y)$ whose roots we are interested in finding explicitly.

The other common technique in our proofs is the classical NI, which starts with just one field root, say $\mu_1$ of $\tilde{f}(\bar{0}, y)$, and builds the full power series on it. Let $E \subseteq \overline{\mathbb{F}}$ be the smallest field where a root $\mu_1$ can be found. Say, $g(\tilde{f}(\bar{0}, y)$ is the minimal polynomial for $\mu_1$. 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 [vzGG13]. In circuits, it means that we make $\deg(E/\mathbb{F})$ copies of each gate and simulate the algebraic operations on these 'tuples' following the $\mathbb{F}$-module structure of $E[\overline{x}]$.

Once we have found all the power series roots of $\tilde{f}(\overline{x}, y)$ over $E[\overline{x}, y]$, say starting from each of the conjugates $\mu_1, \ldots, \mu_k \in E$, 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 $\tilde{f}(\bar{0}, y)$. A more practical method is: While solving the linear system over $E$ in $s$-7 (Algorithm in Theorem 3) we can demand an $E$-solution $u$. Basically, at the level of the algorithm in Lemma 22, we can rewire the linear system $Mw = (\sum_{0 \leq i \leq d} M_i \cdot x_i) \cdot w = 0$ as $M_i w = 0$ ($i \in \{0, d\}$), where the entries of the matrix $M_i$ are given as formulas (respectively ABPs) computing a $\poly(n)$ degree polynomial in $E[\overline{x}]$. This way we get the desired $E$-solution $u$. Then, $s = 8$-9 will yield an irreducible polynomial factor of $f$ in $E[\overline{x}, y]$. This sketches the following more practical version of Theorem 3.

Theorem 26. For $\mathbb{F}$ a number field, a local field, or a finite field (with characteristic $> \deg(f)$), there exists a randomized $\poly(n) \cdot \log(n)$-time algorithm that: for a given $\poly(n^{\log n})$ size formula (respectively ABP) $f$ of poly($n$)-degree and bitsize $s$, outputs $\poly(n^{\log n})$ sized formulas (respectively 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 [vzGG13, Part III] & [Pau01]. See [Gao03] for multivariate factoring over the algebraic closure of a field.

The allRootsNI method in Theorem 1 seems to require all the roots $\mu_i, i \in \{d_0\}$, to begin with. Let $\tilde{u}_1 := \text{rad}(u_1(\overline{\tau}))$. Since $\mu_i$'s are in the splitting field $E \subset \overline{\mathbb{F}}$ of $\text{rad}(\tilde{u}_1(\bar{0}, y))$, 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 $\tilde{h}_1 := h_1(\overline{\tau})$ of $\text{rad}(\tilde{u}_1)$ is some product of these $(y - g_i^{\leq d_0})'$s mod $p^{d_i+1}$. So, for the polynomial $\tilde{h}_1$ in $E[\overline{x}, y]$ 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).
6.3 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 the 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 6.2).

When \( p \) is smaller than the degree of the input polynomial in Theorem 3, adapting an idea from [KSS15, Section 3.1], we claim that we can give \( n^{O(\lambda \log n)} \)-sized formula (respectively ABP) for the \( p^{\ell_i} \)-th power of \( f_i \), where \( f_i \) is a factor of \( f \) whose multiplicity is divisible exactly by \( p^{\ell_i} \), and \( \lambda \) is the number of distinct \( p \)-powers that appear.

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

Theorem 3 can be extended to all characteristics as follows.

**Theorem 27.** Let \( \mathbb{F} \) be of characteristic \( p \geq 2 \). Suppose the \( \text{poly}(n) \)-degree polynomial given by a \( n^{O(\log n)} \) size formula (respectively ABP) factors into irreducibles as \( f(\overline{x}) = \prod_{i} f_i^{p^{\ell_i}}, \) where \( p \nmid \ell_i \). Let \( \lambda := \# \{ \ell_i | \} \).

Then, there is a \( \text{poly}(n^{\lambda \log n}) \)-size formula (respectively ABP) computing \( \hat{f}^{p^\lambda} \) over \( \mathbb{F}_p \).

**Proof sketch.** Note that \( \lambda = O(\log_p n) \).

Let the transformed polynomial of degree \( d \) split into power series roots as follows: \( \hat{f} := f(x, y) = \prod_{i} (y - g_i)^{\gamma_i} \), where \( p \nmid \gamma_i \). Let \( \gamma_i := \# \{ \ell_i | \} \).

Then, there is a \( \text{poly}(n^{\lambda \log n}) \)-size formula (respectively ABP) computing \( \hat{f}^{p^\lambda} \) over \( \mathbb{F}_p \).

Let the transformed polynomial of degree \( d \) split into power series roots as follows: \( \hat{f} := f(x, y) = \prod_{i} (y - g_i)^{\gamma_i} \), where \( p \nmid \gamma_i \). Let \( \gamma_i := \# \{ \ell_i | \} \).

Then, there is a \( \text{poly}(n^{\lambda \log n}) \)-size formula (respectively ABP) computing \( \hat{f}^{p^\lambda} \) over \( \mathbb{F}_p \).

**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 \( \gamma_i \not\equiv 0 \mod p \).

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

**Theorem 28.** Let \( \mathbb{F} \) be of characteristic \( p \geq 2 \). Let \( f = u_0 u_1 \) such that \( \text{size}(f) + \text{size}(u_0) \leq s \). Any factor of \( \text{rad}_p(u_1) \) has size \( \text{poly}(s + \deg(\text{rad}_p(u_1))) \) over \( \mathbb{F} \).
Proof idea: Observe that the roots with multiplicity divisible by $p$ do not contribute to the allRootsNI process. So, the process works with $\text{rad}_p(u_1)$, and the linear algebra complexity involved is polynomial in its degree.

7 CONCLUSION

We analyzed the complexity of approximating power series roots (up to some degree) of a multivariate polynomial in various models. As the roots are related to factors, we get results on polynomial factoring in various models as well.

Finally, we list a few open questions related to multivariate factoring.

(1) The Factor Conjecture states that for a nonzero polynomial $f: g | f \implies \text{size}(g) \leq \text{poly}(\text{size}(f), \deg(g))$.

Motivated by Theorem 1, we would like to strengthen the Factor conjecture to a conjecture about the squarefree part of a circuit:

Conjecture 1 (Radical Conjecture). For a nonzero $f: \min\{\deg(\text{rad}(f)), \text{size}(\text{rad}(f))\} \leq \text{poly}(\text{size}(f))$.

Is the above conjecture true if we replace size by size?

(2) Suppose we have a circuit with division gates computing a polynomial of degree $d$. Can we get a circuit of size $\text{poly}(s)$ computing the same polynomial without using any division gate [Kal87]? A positive answer to this question would prove the Factor conjecture as a corollary. Strassen’s classic result gives a $\text{poly}(s, d)$ bound for this problem. Recently, [DJPS21] showed that division elimination can be efficiently done if the divisor is a low degree polynomial (computed by a small circuit).

(3) Given two polynomials computed by circuits of size $s$, can we get a circuit computing their gcd in size $\text{poly}(s)$? Kaltofen [Kal87] gave $\text{poly}(s, d_g)$ size upper bound, where $d_g$ is the degree of the gcd.

(4) Is VF closed under factoring? We might consider Theorem 3 as a positive evidence. Additionally, a special case when $f = g^e$ for some irreducible polynomial $g$, can be solved. This is easy to see using the classic Taylor series of $(1 + f)^{1/e}$, where $f \in \langle x \rangle$.

In fact, what about the classes which are contained in $VF(n^{\log n})$ but larger than $VF$. For example, is $VF(n^{\log \log n})$ closed under factoring?

(5) Can we compute factors of polynomials computed by circuits of low depth by keeping the depth constant and the size small? To further motivate this question, we mention the recent breakthrough by Limaye, Srinivasan and Tavenas [LST21] where the authors gave the first superpolynomial lower bound against constant depth circuits. They also gave the first subexponential PIT for the same model using an arithmetic hardness vs randomness result from [CKS19b]. The work of [CKS19b] needed a size upper bound of roots of low depth polynomials (different from the related prior work by [DSY09]). The application of [CKS19b] in [LST21] shows the importance of studying factoring and root finding for restricted classes to settle fundamental questions in algebraic complexity.

Finally, our results weaken when the underlying field $\mathbb{F}$ is not algebraically closed or has a small prime characteristic (Sections 6.2, 6.3). Can we strengthen the methods to work for all $\mathbb{F}$?

Acknowledgements. We thank Rafael Oliveira for extensive discussions regarding his works and circuit factoring in general. In particular, we used his suggestions about VNP and $\overline{\text{VP}}$ in our results. We thank Manindra Agrawal, Sumanta Ghosh, Partha Mukhopadhyay, Thomas Thierauf, and Nikhil Balaji for helpful discussions. 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, Google India Research Program Team for the support of all travel
expenses and PhD Fellowship. N.S. thanks for the funding support from DST (DST/SJF/MAA-01/2013-14), DST-SERB (CRG/2020/000045) and N. Rama Rao Chair. A.S. would like to thank Microsoft Research Lab India, IARCS, and ACM India for supporting with travel grants for conferences. Previously, A.S. was supported by MHRD (Govt of India) graduate student fellowship. Currently, he is supported by DFG grant TH 472/5-1.

REFERENCES


Discovering the roots: Uniform closure results for algebraic classes under factoring

[LLMP90] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. The number field sieve. In


Perspectives in Computational Complexity


[Len83] Arjen K Lenstra. Factoring polynomials over algebraic number fields. In


Manuscript submitted to ACM
A numerical analog of Theorem 1: proof of Claim 1

Claim 1 (restated). For each root $a \in (0, 1)$ of $f(x)$, there is some $2^m$-bit approximation $a'$ such that $\text{bitsize}(a') \leq O((s + m) \cdot \log\left(\frac{1}{\epsilon}\right))$, where $\text{bitsize}(f) = s$, and $\epsilon \in (0, 1)$ lower bounds the gap between $a$ and the other roots of $f(x)$.

Proof sketch: Now we give the main $s$ in the numerical result (the algebraic analog Theorem 1 would be more involved to prove as it is stronger). We use general Newton iteration (or, NI with multiplicity). Observe that, we are interested in an existential statement; so, assume that we know the multiplicity $\gamma$ of the root $a$ before-hand. Suppose, $f(x) = (x-a)^{\gamma}g(x)$ where $g(a) \neq 0$. Assume that $g(x) = \prod_{i=1}^{\gamma}(x-b_i)^{y_i}$.

Hypothesis says that $|a - b_i| \geq \epsilon$, for all $i$. We will use general NI to approximate $2^m$ bits of $a$ (in this case after the decimal-point only). We will start with $y_0$ such that $|y_0 - a| \leq \epsilon \cdot 2^{-3(m+s) - 1}$. Thus, to start with, $y_0$ has a trivial circuit of bitsize $O((m + s) \log\left(\frac{1}{\epsilon}\right))$.

Next, we use the general NI formula [DB08, Eqn.6.3.13], i.e.

$$y_{t+1} = y_t - \gamma \frac{f}{f'}|_{y_t}.$$

Finally, we need to show that the process has quadratic convergence. Inductively, we want to show that for all $t \geq 0$,

$$|y_t - a| \leq \epsilon \cdot 2^{-3(m+s) - 2t}.$$

Note that the above inequality implies that $y_m$ is a $2^m$-bit approximation of $a$. Moreover, computing $y_{t+1}$ as a circuit—given the value $y_t$ and circuits for $f(x), f'(x)$—requires $O(s)$ additional bitsize (remember $\div$ is allowed in the circuit).

So, $y_m$ will have a circuit of bitsize $O((s + m) \log\left(\frac{1}{\epsilon}\right))$.

Manuscript submitted to ACM
We are only left to prove Equation 5. We have,

\[
\frac{f(y_t)}{f'(y_t)} = \frac{(y_t - a)y'g(y_t)}{(y_t - a)^r \cdot (yg(y_t) + (y_t - a)g'(y_t))} = \frac{yg(y_t) + (y_t - a)g'(y_t)}{(y_t - a)g(y_t)}.
\]

Hence,

\[
|y_{t+1} - a| = \left| y_t - a - \frac{f(y_t)}{f'(y_t)} \right|
\]

\[
= \left| (y_t - a) \left( 1 - \frac{yg(y_t)}{yg(y_t) + (y_t - a)g'(y_t)} \right) \right|
\]

\[
= |y_t - a|^2 \frac{g'(y_t)}{yg(y_t) + (y_t - a)g'(y_t)}
\]

\[
= |y_t - a|^2 \frac{yg(y_t) + (y_t - a)g'(y_t)}{yg(y_t) + (y_t - a)g'(y_t)} -1
\]

\[
\leq |y_t - a|^2 \frac{yg(y_t)}{yg(y_t) - (y_t - a)} -1
\]

Observe that by Leibniz rule:

\[
g'(y_t) \frac{r}{g(y_t)} = \sum_{i=1}^{r} \frac{y_i}{y_t - b_i}
\]

By the induction hypothesis \(|y_t - a| \leq \epsilon 2^{-3r + s + m} \cdot 2^{-t'} \); so, we have for all \(i \in [r] \):

\[
|y_t - b_i| \geq |a - b_i| - |y_t - a| \geq \epsilon \cdot (1 - 2^{-t'} 2^{m}).
\]

Since \(y_t, r \leq 2^s \) (because bitsize-s circuit can have degree at most \(2^s \)), we can upper bound as:

\[
\left| \frac{g'(y_t)}{g(y_t)} \right| \leq \sum_{i \in [r]} \frac{y_i}{y_t - b_i} \leq \frac{2^{2s}}{\epsilon (1 - 2^{-t'} 2^{m})}.
\]

This means \( \left| \frac{g'(y_t)}{g(y_t)} \right| \geq 2^{-2s-1} \epsilon \). Consequently,

\[
\left| \frac{g(y_t)}{g'(y_t)} - |y_t - a| \geq 2^{-2s-1} \epsilon - \epsilon 2^{-3s} \cdot 2^{-t'}
\]

\[
= \epsilon 2^{-3s} \cdot \left( 2^{-1} - 2^{m} \cdot 2^{-t'} \right)
\]

Hence,

\[
|y_{t+1} - a| \leq |y_t - a|^2 \left| \frac{g(y_t)}{g'(y_t)} - |y_t - a| \right|^{-1}
\]

\[
\leq 2^{-2s+1} \left( \epsilon \cdot 2^{-3s} \right)^2 \cdot \left( \epsilon 2^{-3s} \right)^{-1}
\]

\[
\leq 2^{-2s+1} \cdot \epsilon \cdot 2^{-3s(m+s)}
\]

This finishes the inductive step and we are done. \( \square \)

We leave some interesting questions open: Can we improve bitsize\((a)\) to poly\((s + m)\)? Can we prove a bound for bitsize\((a)\) without requiring \(\div\) gates?