Electronic Colloquium on Computational Complexity, Report No. 171 (2025)

Modular composition & polynomial GCD in the *border* of small, shallow circuits

Robert Andrews* Mrinal Kumar† Shanthanu S. Rai†

Abstract

Modular composition is the problem of computing the coefficient vector of the polynomial $f(g(x)) \bmod h(x)$, given as input the coefficient vectors of univariate polynomials f, g, and h over an underlying field $\mathbb{F}$. While this problem is known to be solvable in nearly-linear time over finite fields due to work of Kedlaya & Umans, no such near-linear-time algorithms are known over infinite fields, with the fastest known algorithm being from a recent work of Neiger, Salvy, Schost & Villard that takes $O(n^{1.43})$ field operations on inputs of degree n. In this work, we show that for any infinite field $\mathbb{F}$, modular composition is in the *border* of algebraic circuits with division gates of nearly-linear size and polylogarithmic depth. Moreover, this circuit family can itself be constructed in near-linear time.

Our techniques also extend to other algebraic problems, most notably to the problem of computing greatest common divisors of univariate polynomials. We show that over any infinite field $\mathbb{F}$, the GCD of two univariate polynomials can be computed (piecewise) in the border sense by nearly-linear-size and polylogarithmic-depth algebraic circuits with division gates, where the circuits themselves can be constructed in near-linear time. While univariate polynomial GCD is known to be computable in near-linear time by the Knuth–Schönhage algorithm, or by constant-depth algebraic circuits from a recent result of Andrews & Wigderson, obtaining a parallel algorithm that simultaneously achieves polylogarithmic depth and near-linear work remains an open problem of great interest. Our result shows such an upper bound in the setting of border complexity.

*Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada. Part of this work was supported by the Simons Institute for the Theory of Computing, and was conducted when the author was visiting the Institute. Email: randrews@uwaterloo.ca.

†Tata Institute of Fundamental Research, Mumbai, India. Email: {mrinal, shanthanu.rai}@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project number RTI400112, and partially supported by an SERB grant, a Premji Invest grant and TCS PhD Fellowship of the third author.

ISSN 1433-8092

Contents

1 Introduction

In this work, we study the complexity of two natural algebraic problems: modular composition and greatest common divisors (GCDs) of univariate polynomials. We start with a brief discussion of these problems and the state of the art for them.

Modular composition: For modular composition, the input consists of the coefficient vectors of three univariate polynomials $f(x)$, $g(x)$, and $h(x)$ over an underlying field $\mathbb{F}$, and the goal is to compute the coefficient vector of the remainder obtained by dividing the composed polynomial $f(g(x))$ by $h(x)$. While several natural algebraic problems such as polynomial multiplication, multipoint evaluation, polynomial interpolation, and division with remainder have nearly-linear-time algorithms,[1] all building upon the remarkable fast Fourier transform, obtaining such nearly-linear-time algorithms for modular composition appears to be a much more challenging problem. In fact, even the seemingly easier task of obtaining an algorithm that is faster than the obvious $O(n^2)$ time algorithm for modular composition doesn't appear to have an immediate solution!

In 1978, Brent and Kung [BK78] gave an algorithm for modular composition that runs in time $\widetilde{O}(n^{(\omega+1)/2})$, where ω is the exponent for matrix multiplication. In their algorithm, they used a *baby-step giant-step* based technique to reduce the problem to rectangular matrix multiplication. Since we have matrix multiplication algorithms with $\omega < 3$, this gives us a non-trivial sub-quadratic algorithm for modular composition. But even after assuming the best matrix multiplication constant of $\omega = 2$, their algorithm does not beat the runtime of $O(n^{1.5})$. For many years, the algorithms from [BK78] essentially remained the fastest algorithm for modular composition over any field $\mathbb{F}$. Over finite fields, this state of art was significantly improved in a work of Kedlaya & Umans [KU08] who gave a nearly linear time algorithm for modular composition over such fields. The techniques in [KU08] appear to be heavily tailored to the setting of finite fields and it is unclear if the ideas from there can be lifted to obtain speed ups over fields like the complex numbers. More recently, a work of Neiger, Salvy, Schost & Villard [NSSV24] gave the first algorithm for modular composition with time complexity better than $n^{1.5}$ over infinite fields. Their algorithm, which is randomized, again involves (an extremely delicate) reduction to matrix multiplication, and is the first improvement over the results in [BK78] over infinite fields. As noted in [NSSV24], the algorithm there does not yield a near-linear-time algorithm for modular composition, even if one assumes the exponent of matrix multiplication ω is 2. Over finite fields, while the algorithm in [NSSV24] is much slower than that in [KU08], it does have a qualitative advantage—it can be viewed as being *algebraic* over the underlying field, i.e., it only uses basic arithmetic over the underlying field and zero tests.[2]

[1] Throughout the paper, by *nearly-linear time*, we mean time complexity of the form $n^{1+o(1)}$. We also use the notation $\widetilde{O}(n)$ to denote functions bounded by $O(n \operatorname{poly}(\log n))$.

[2] Such algorithms are referred to as *algebraic computation trees* in the literature, and can essentially be thought of as algebraic circuits empowered with the ability to branch on a zero test. See Chapter 4 in [BCS97] for a detailed discussion on these models.

This is in contrast to the algorithm in [KU08], which makes crucial use of bit operations on the field elements. Recently, there has been significant progress for the special case of modular composition where $h(x) = x^n$, which is also referred to as power series composition. Kinoshita and Li [KL24] gave a nearly-linear-time algorithm for power series composition based on the classical technique of Graeffe iteration. Their algorithm only uses algebraic operations over the underlying field and works over all fields.

The scientific interest in the problem of designing faster algorithms for modular composition stems from two sources. The first source of motivation is the numerous applications and connections that fast algorithms for modular composition have to fast algorithms for other algebraic problems. This includes applications towards obtaining the current fastest algorithms for factorization of univariate polynomials over finite fields [KS98, KU08], algorithms for normal bases computation [KS98, GJS21], arithmetic operations on algebraic numbers [BFSS06], and computing minimal polynomials of algebraic numbers [Sho94, Sho95, Sho99]. The second source of motivation is the fact that, unlike many other basic problems in computational algebra, modular composition seems resistant to the design of near-linear-time algorithms. Numerous problems have decades-old near-linear-time algorithms that use ideas based on the fast Fourier transform, but there were no linear-time algorithms for modular composition over any field till the work of Kedlaya & Umans. The fastest algorithms for modular composition over infinite fields remain far from near-linear time, and designing such algorithms remains an open problem of great interest (see, e.g., open problem 12.19 in [vzGG13] and open problem 2.4 in [BCS97]).

Greatest common divisors of polynomials: The second main problem of interest in this work is the task of computing the greatest common divisor (GCD) of two univariate polynomials with coefficients in a field. Computing the GCD is a fundamental operation in computer algebra, and the complexity of this task has been studied extensively. The Euclidean algorithm, one of the oldest algorithms to survive to the modern day, computes the GCD of two degree-n polynomials in $\widetilde{O}(n^2)$ time when implemented with fast polynomial arithmetic. The Knuth–Schönhage algorithm—also known as the half-GCD algorithm—improves this to $\widetilde{O}(n)$ time by a clever use of divide-and-conquer [BCS97, Chapter 3]. The GCD can also be computed quickly in parallel: using parallel algorithms for linear algebra, Borodin, von zur Gathen, and Hopcroft [BvH82] showed that the GCD can be computed in $O(\log^2 n)$ parallel time with $\mathrm{poly}(n)$ work.[3] A surprising recent result of Andrews & Wigderson [AW24] improved this to $O(\log n)$ parallel time over fields of zero or sufficiently large characteristic, and subsequent work of Bhattacharjee et al. [BKR+25] extended this to all sufficiently large fields. In fact, these later results show that the GCD can be computed (piecewise) by unbounded fan-in algebraic circuits of constant depth and polynomial size.

[3] As with modular composition, we formalize algorithms for the GCD using the model of algebraic computation trees. Here, the parallel time of an algorithm corresponds to the depth of the tree, and the total work corresponds to the number of gates in the tree.

While the GCD can be computed either in near-linear time sequentially, or in $O(\log n)$ parallel time, it is not clear if there is a single algorithm that simultaneously achieves polylogarithmic parallel time and near-linear total work. The divide-and-conquer scheme appearing in the half-GCD algorithm is not obviously parallelizable, since the input to one recursive subproblem depends on the solution of its sibling subproblem. Known parallel algorithms either make use of linear algebra on $n \times n$ matrices or interpolate coefficients of polynomials of degree n, both of which incur at least $O(n^2)$ total work when implemented in a straightforward manner. Finding such an algorithm for the GCD is an interesting and important challenge.

Having discussed the main motivating questions for this work, we are now ready to state our results.

2 Our results and prior work

To state our results, we need the notion of algebraic circuits and border complexity of rational functions over an underlying field $\mathbb{F}$. We start by recalling these notions.

Algebraic circuits and border complexity

An algebraic circuit over a field $\mathbb{F}$ is a directed acyclic graph with leaves labeled by formal variables and field constants, and internal nodes (called gates) labeled by field operations $(+, \times, \div)$. For this work, we will consider circuits where the internal gates have fan-in 2. The circuits compute a formal rational function over the underlying field in a natural sense—an input gate computes the polynomial equal to the field constant or the variable that is its label; a sum $(+)$ or product $(\times)$ gate computes the sum or product, respectively, of its inputs; and a division gate $(\div)$ gate outputs the rational function whose numerator is the gate's left child and whose denominator is the right child. The size of such a circuit refers to the number of edges in it and the depth refers to the length of the longest path from an output gate to an input gate.

We say that a polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ is in the *border* of algebraic circuits of size s if there is an algebraic circuit C of size s over the field $\mathbb{F}(\varepsilon)$, for a new formal variable ε, such that the polynomial computed by C is of the form $f(\mathbf{x}) + \varepsilon g(\mathbf{x}, \varepsilon)$, where $g \in \mathbb{F}[\varepsilon][\mathbf{x}]$ is a polynomial in $\mathbf{x}$ and ε. We abbreviate this by saying that C computes $f(\mathbf{x}) + O(\varepsilon)$. In other words, if we were allowed to set ε to zero in C, the resulting circuit would compute the polynomial f. However, the circuit C may not be well-defined at $\varepsilon = 0$, since C is permitted to divide by ε during its computation. While the border complexity of a polynomial is clearly upper bounded by its algebraic circuit complexity, the relationship in the other direction is not well understood and is an important and active research direction in algebraic and geometric complexity [GMQ16, BIZ18, Kum20, DDS22, IS22, CGGR23, DGI+24, DIK+24, Shp25]. For more on this, we refer the reader to the recent survey of Dutta and Lysikov [DL25].

We now state and discuss our results. Throughout, we use $\mathrm{coeff}(f)$ to refer to the coefficient vector of a polynomial f.

Border complexity of modular composition

As our first main result, we show that over any infinite field $\mathbb{F}$, if we view modular composition as a formal rational function in the coefficients of the three input polynomials of degree n, then its *border algebraic circuit complexity* (henceforth simply border complexity) is nearly linear in n. Moreover, these near-linear-size circuits are of polylogarithmic depth and are uniform in the sense that they can be constructed in near linear time in their size. More formally, we have the following theorem.

Theorem 2.1 (Border complexity of modular composition)**.** *Let $\mathbb{F}$ be an infinite field and ε be a formal variable. There is a family $\{C_n\}_{n\in\mathbb{N}}$ of multi-output algebraic circuits with division gates, defined over the field $\mathbb{F}(\varepsilon)$, such that C_n has size $\widetilde{O}(n)$, depth $\operatorname{polylog} n$, and for all polynomials $f, g, h \in \mathbb{F}[x]$ of degree equal to n, we have*

$$C_n(\mathrm{coeff}(f), \mathrm{coeff}(g), \mathrm{coeff}(h)) = \mathrm{coeff}(f(g(x)) \bmod h(x)) + O(\varepsilon).$$

Moreover, there is an algorithm that, given n as input, outputs a description of C_n in time $\widetilde{O}(n)$.

A classical theorem of Strassen [Str73] shows that division gates can be eliminated from algebraic circuits that compute polynomials, up to a polynomial (in degree) blow up in size. The coefficient vector $\mathrm{coeff}(f(g(x)) \bmod h(x))$ is a polynomial function of the coefficients of f, g, and h if we constrain h to be a monic polynomial, so in principle one could eliminate the use of division gates from the circuits appearing in Theorem 2.1. However, it is not clear if this division elimination can be implemented in Theorem 2.1 (even for monic h) in the context of the results in this work, since we can't even tolerate linear blow up in the circuit size in the process.

Border complexity of polynomial GCD

Our second main result establishes that, over any infinite field $\mathbb{F}$, the GCD of univariate polynomials is (piecewise) in the border of circuits of near-linear size and polylogarithmic depth. As in Theorem 2.1, these circuits are uniform. More formally, we have the following theorem.

Theorem 2.2 (Border complexity of GCD)**.** *Let $\mathbb{F}$ be an infinite field and ε be a formal variable. There is a family $\{C_{n,d}\}_{n,d\in\mathbb{N}}$ of multi-output algebraic circuits, with division gates and constants from the field $\mathbb{F}(\varepsilon)$, such that $C_{n,d}$ has size $\widetilde{O}(n)$, depth $\operatorname{polylog} n$, and for all polynomials $f(x), g(x) \in \mathbb{F}[x]$ of degree n such that $\gcd(f, g)$ has degree d, we have*

$$C_{n,d}(\mathrm{coeff}(f), \mathrm{coeff}(g)) = \mathrm{coeff}(\gcd(f, g)) + O(\varepsilon).$$

Moreover, there is an algorithm that, given n, d as input, outputs $C_{n,d}$ in time $\widetilde{O}(n)$.

The parameterization of the circuit family by both the degree of the input polynomials and the degree of the GCD of the inputs might seem a little unusual in Theorem 2.2. However, this is necessary, since the coefficient vector of $\gcd(f, g)$ is not a formal rational function of the coefficients of f, g, so we cannot hope to compute it using a single algebraic circuit (even in the border setting). However, once we fix the degree of the GCD and consider inputs f, g that have GCD of this fixed degree, the coefficient vector of $\gcd(f, g)$ is indeed a rational function of the coefficients of f, g. This classical fact has an elementary proof using Bézout's identity (see Section 9).

It would be interesting to understand if the degree of the GCD of two polynomials can itself be computed in near linear time and polylog depth. As of now, our proof of Theorem 2.2 does not appear to yield such a subroutine.

Border complexity of symmetric polynomials

At the heart of our proofs of Theorem 2.1 and Theorem 2.2 is perhaps an independently natural and interesting technical result about the complexity of symmetric polynomials. Recall that from the fundamental theorem of symmetric polynomials, for every symmetric n-variate polynomial $P(\mathbf{x})$, there exists a unique n-variate polynomial Q such that

$$P(\mathbf{x}) = Q(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x})),$$

where $\mathbf{Esym}_i(\mathbf{x})$ is the elementary symmetric polynomial of degree equal to i in the variables $\mathbf{x}$. From a computational perspective, it is interesting to understand how the algebraic complexities of P and Q relate to each other. Since elementary symmetric polynomials have small constant-depth algebraic circuits (over all sufficiently large fields), we immediately have that the algebraic circuit complexity of P is *not much larger* than the circuit complexity of Q. However, the relation in the other direction is nontrivial and much more interesting.

A significant step towards understanding this problem is a work of Bläser & Jindal [BJ19], who showed that the algebraic circuit complexity of Q is at most polynomially bounded in the algebraic circuit complexity of P. As it is, this result did not appear to extend to weaker sub-classes of algebraic circuits, for instance, algebraic formulas or constant-depth circuits, since this structure did not appear to be preserved in the transformation in the proof. Such an extension was shown in a recent work of Bhattacharjee, Kumar, Ramanathan, Rai, Saptharishi & Saraf [BKR+25], who showed that this polynomial equivalence in the complexities of P and Q also extends to their formula complexity and constant-depth circuit complexity. The polynomial gap in the algebraic circuit complexity of P and Q shown in results in [BJ19] and [BKR+25] turns out to be insufficient for our applications in this paper. The main technical question of interest is to understand whether the circuit complexity of Q is nearly linear in the circuit complexity of P. While we are unable to

answer this question in general, we show the following weaker result that turns out to be sufficient for our applications towards Theorem 2.1 and Theorem 2.2.

Theorem 2.3 (Border Complexity of symmetric polynomials)**.** *Let $\mathbb{F}$ be a field of size at least n. Let $P(\mathbf{x})$ be a symmetric polynomial that is computable by a circuit of size s and depth Δ, and let $Q(\mathbf{z})$ be the unique polynomial such that $P(\mathbf{x}) = Q(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x}))$.*

If $Q(\mathbf{z})$ is a homogeneous polynomial of degree d, then $Q(\mathbf{z})$ is computable in the border of a circuit of size $\widetilde{O}(s + n \log d)$ and depth $\Delta + \mathrm{polylog}(n, d)$. In other words, there is a circuit of this size and depth, defined over the field $\mathbb{F}(\varepsilon)$, that computes $Q(\mathbf{z}) + O(\varepsilon)$.

A simple and clean application of the technical ideas in this work is the following special case of computing the resultant of two bivariate polynomials: given two univariate polynomials $f, g \in \mathbb{F}[x]$, compute the resultant $\mathrm{Res}_x(y + f(x), g(x))$, where y is a fresh variable. This resultant is a univariate polynomial in y, so the goal is to compute the coefficients of this polynomial using arithmetic operations from the base field $\mathbb{F}$. We show that this resultant is in the border of nearly linear size circuits of polylogarithmic depth. More formally, we have the following theorem.

Theorem 2.4 (Resultant of $y + f(x)$ and $g(x)$)**.** *Let $\mathbb{F}$ be an infinite field and ε be a formal variable. There is a family $\{C_n\}_{n \in \mathbb{N}}$ of multi-output algebraic circuits with division gates, defined over the field $\mathbb{F}(\varepsilon)$, such that C_n has size $\widetilde{O}(n)$, depth $\mathrm{polylog}\, n$, and for all polynomials $f, g \in \mathbb{F}[x]$ of degree equal to n, we have*

$$C_n(\mathrm{coeff}(f), \mathrm{coeff}(g)) = \mathrm{coeff}(\mathrm{Res}_x(y + f(x), g(x))) + O(\varepsilon).$$

Moreover, there is an algorithm that, given n as input, outputs C_n in time $\widetilde{O}(n)$.

Apart from demonstrating the main ideas in the proofs of Theorem 2.1 and Theorem 2.2, computing $\mathrm{Res}_x(y + f(x), g(x))$ is an interesting standalone problem in its own right. This resultant is precisely the characteristic polynomial of the linear map $h \mapsto -f \cdot h$ in the quotient ring $\mathbb{F}[x]/(g(x))$ (see, e.g., [CLO05, Chapter 4, Proposition 2.7]). Just like modular composition, no near-linear-time algorithm is known for computing the characteristic polynomial in univariate quotient rings. The fastest-known algorithm for this problem is due to Bostan, Flajolet, Salvy, & Schost [BFSS06] and uses $O(n^{\frac{\omega+1}{2}})$ arithmetic operations, where ω is the exponent of matrix multiplication.

2.1 Our results in the context of prior work

Having stated our main results in the previous section, we now briefly discuss their relation to prior work.

Most of the research on proving upper bounds on the complexity of modular composition and polynomial GCD has been from the point of view of designing (uniform) algorithms for them. This includes algebraic algorithms where the complexity is measured in terms of the number of arithmetic operations (e.g., [BK78, NSSV24]), as well as (heavily) non-algebraic algorithms where

bit access to the input field elements is used (perhaps the most striking example being the work of Kedlaya & Umans [KU08]). Especially over infinite fields, seeking algebraic algorithms (that can be viewed as algebraic circuits, perhaps accompanied with branching on zero tests) is an extremely natural question. However, in spite of decades of interest, these problems continue to be wide open. As discussed in the introduction, while modular composition is solvable in nearly-linear time over finite fields from the work of Kedlaya & Umans [KU08], the fastest algorithm over infinite fields is from the recent work of Neiger et al. [NSSV24] and runs in time $O(n^{1.43})$, where n is the degree of the input polynomials. The results in [NSSV24] were essentially the first substantial improvement on the state of the art for this problem since the work of Brent & Kung [BK78] from the 1970s. Similarly, while polynomial GCD is known to be computable both in near-linear time via the half GCD algorithm, and via log-depth circuits from the works of Andrews & Wigderson [AW24] and Bhattacharjee et al. [BKR$^+$25], it is not known how to combine these guarantees together.

Given the slow progress on these problems in the preceding decades, it seems very natural to relax our requirements and seek non-uniform upper bounds on the complexity of these problems. The results in this paper were motivated by this goal. While we do not manage to answer these questions to our satisfaction, the results in the paper represent some encouraging progress towards this and hopefully offer some interesting insights into the nature of these questions. The upper bounds proved here are all in terms of border algebraic circuit complexity, but modulo this (fairly non-trivial) caveat, the results are precisely of the nature that we would have sought—the circuits for modular composition and polynomial GCD are both nearly-linear size and polylogarithmic depth. In addition to this, these circuits are uniformly constructible in nearly-linear time!

While the results in this paper are among the first upper bounds for modular composition and polynomial GCD in terms of their border circuit complexity, the use of border complexity as a tool for interesting upper bounds for algebraic problems is hardly new or surprising. For example, border complexity figures prominently in the study of matrix multiplication. Border rank was introduced as a new tool in the design of matrix multiplication algorithms by Bini, Capovani, Romani, & Lotti [BCRL79] (see also [Bin80]), and since then, essentially all algorithmic progress has proceeded by obtaining better upper bounds on the border tensor rank of the matrix multiplication tensor. The border complexity of algebraic problems has also been studied extensively under the name *any precision approximation algorithms*, or APA-algorithms. For more discussions on such algorithms, we refer the reader to [BP94, Chapter 3, Section 8; and Chapter 4, Section 2, Table 2.1] and the references therein. Another result of a very similar flavor to those in this work is an algorithm of Bini [Bin84] to invert triangular Toeplitz matrices. In more recent decades, border complexity was used in a work of Bürgisser [Bür04], who showed a polynomial upper bound on the border complexity of *low* degree factors of multivariate polynomials with exponentially large degree but computable by algebraic circuits of small size. The question of extending this border complexity upper bound to the non-border setting is the so-called *factor conjecture* and remains an

extremely interesting open problem. Yet another surprising upper bound in the setting of border complexity is a result of Bringmann, Ikenmeyer & Zuiddam [BIZ18], who showed that over any field of characteristic different from 2, any polynomial computable by a size s formula can be computed in the border sense by a width 2 algebraic branching program of size $\text{poly}(s)$. While a non-border version of such a result is known for width-3 algebraic branching programs (see Ben Or & Cleve [BC88]), it is also known that such a non-border upper bound is false for width-2 algebraic branching programs [AW16].

Despite the utility of border complexity as an algorithmic tool, we still do not understand how it compares to the standard notion of algebraic circuit complexity. One of the primary motivations for studying border complexity comes from the fact that almost all known lower bounds for algebraic circuit classes are proven using continuous methods—often using arguments based on matrix rank—and so extend immediately to the border variant of the corresponding circuit class. Because of this, it is important to understand border complexity, as it provides a sanity check on what lower bounds one could hope to prove using existing techniques. These techniques are captured by the notion of an algebraically natural proof, and we refer the reader to [FSV18, GKSS17] for further discussion on the role such proofs play in algebraic complexity.

Because the relationship between border circuit complexity and (non-border) algebraic complexity or algorithms is poorly-understood, there are two different conclusions that one may draw from our results. For the optimist, the results in this paper can be seen as evidence that modular composition can be solved in nearly-linear time, and similarly that the polynomial GCD can be computed in near-linear time and polylogarithmic depth. For the pessimist who thinks such algorithms do not exist, our results offer a technical barrier: any proof of an $\Omega(n^{1+\varepsilon})$ lower bound on the complexity of modular composition cannot be algebraically natural in the sense of [FSV18, GKSS17]. Such a proof would imply the same lower bound on the border complexity of modular composition, which stands in direct contradiction with the results of this paper. Lower bound techniques in algebraic complexity are far from the point where we could hope to prove an $\Omega(n^{1+\varepsilon})$ lower bound for any problem, let alone modular composition. However, this proof barrier suggests that once such lower bound techniques are developed, they will be inapplicable to problems like modular composition. Likewise, our results establish a barrier against proving that the polynomial GCD cannot be computed in polylogarithmic depth and near-linear total work.

We refrain from further speculation on this and encourage the reader to draw their own conclusions. However, we expect that some of the technical ideas used in the proof of our results, including the relationship to Theorem 2.3 and its analogues, may be of independent interest. We also hope that this work leads to further interest in the border complexity of other natural algebraic problems, especially those that have resisted progress when approached from the perspective of uniform algorithms or classical algebraic circuit complexity.

3 Overview of proofs

In this section, we discuss some of the technical ideas in the proofs of our results. As a warm up to the use of border complexity in this context, we start with a brief sketch of a border version of a recent result of Kinoshita & Li [KL24] who gave a nearly-linear-time algorithm for power series composition (computing $f(g(x))$ modulo x^n). While being elementary in its technical ideas, the algorithm in [KL24] is extremely delicate and carefully crafted in its final details. In contrast, the border upper bound that we discuss here is quite simple, both in terms of high level ideas and their details. While the ideas in this argument do not immediately seem to generalize to the proofs of the main theorems in this work, the simple border upper bound for power series composition was a strong source of motivation.

3.1 Power Series Composition

Given two degree n polynomials $f(x) \in \mathbb{F}[x]$ and $g(x) \in \mathbb{F}[x]$, we would like to compute $f(g(x)) \bmod x^n$. The polynomials are given as a vector in their coefficients, i.e., coeff(f) and coeff(g). Let $\alpha_1, \ldots, \alpha_n$ be n distinct elements in the field $\mathbb{F}$. Let $F(x) := f(g(x)) = F_0 + F_1 x + \ldots F_{n^2} x^{n^2}$. For power series composition, we need to compute $F(x) \bmod x^n = F_0 + F_1 x + \ldots F_{n-1} x^{n-1}$. To this end, we will first compute $F(\varepsilon\alpha_1), \ldots, F(\varepsilon\alpha_n)$ and then extract $F_0, \ldots, F_{n-1}$ from these evaluation (in the border). Note that to compute these coefficients exactly, we need n^2 evaluations, which is way beyond our budget! We build a near-linear-size circuit to compute $F(\varepsilon\alpha_1), \ldots, F(\varepsilon\alpha_n)$ as follows: using univariate multipoint evaluation circuit (see Lemma 4.3), given coeff(g) and $\alpha_1, \ldots, \alpha_n$ as input, we compute $g(\varepsilon\alpha_1), \ldots, g(\varepsilon\alpha_n)$. Again using univariate multipoint evaluation circuit, given coeff(f) and $g(\varepsilon\alpha_1), \ldots, g(\varepsilon\alpha_n)$ as input, we compute $f(g(\varepsilon\alpha_1)), \ldots, f(g(\varepsilon\alpha_n))$.

Observe that $F_0, F_1, \ldots, F_{n-1}$ satisfy the equations

$$\underbrace{\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_2^3 & \cdots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n-1} \end{pmatrix}}_{V} \underbrace{\begin{pmatrix} F_0 \\ \varepsilon F_1 \\ \varepsilon^2 F_2 \\ \vdots \\ \varepsilon^{n-1} F_{n-1} \end{pmatrix}}_{\mathbf{u}} = \underbrace{\begin{pmatrix} F(\varepsilon\alpha_1) \\ F(\varepsilon\alpha_2) \\ F(\varepsilon\alpha_3) \\ \vdots \\ F(\varepsilon\alpha_n) \end{pmatrix}}_{\mathbf{w}} + \underbrace{\begin{pmatrix} O(\varepsilon^n) \\ O(\varepsilon^n) \\ O(\varepsilon^n) \\ \vdots \\ O(\varepsilon^n) \end{pmatrix}}_{\mathbf{e}},$$

where vector $\mathbf{e}$ entries are the "error" terms in ε. So, $\mathbf{u} = V^{-1}\mathbf{w} + V^{-1}\mathbf{e}$. Thus, for $i \in \{0, 1, \ldots, n-1\}$, we have,

$$\varepsilon^i F_i = u_i = (V^{-1}\mathbf{w})_i + (V^{-1}\mathbf{e})_i = (V^{-1}\mathbf{w})_i + O(\varepsilon^n)$$

Using the above observation, we build a near linear size circuit that computes $F_0, F_1, \ldots, F_{n-1}$ in

the border as follows: using univariate interpolation circuit (see Lemma 4.4), given $\alpha_1, \ldots, \alpha_n$ and $F(\varepsilon\alpha_1), \ldots, F(\varepsilon\alpha_n)$ as inputs, we compute $F_0 + O(\varepsilon^n), \varepsilon F_1 + O(\varepsilon^n), \ldots, \varepsilon^{n-1}F_{n-1} + O(\varepsilon^n)$. We divide out the i-th output by ε^i to compute $F_0, F_1, \ldots, F_{n-1}$ in the border.

3.2 Proof of Theorem 2.3

We now discuss the main ideas in the proof of Theorem 2.3 which is one of the main technical ideas driving the results on modular composition and GCD.

Let $P(x_1, \ldots, x_n)$ be a symmetric polynomial. From the fundamental theorem of symmetric polynomials, there exists a unique polynomial $Q(y_1, \ldots, y_n)$ such that $P(x_1, \ldots, x_n)$ equals $Q(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x}))$, where $\mathbf{Esym}_i(\mathbf{x})$ is the elementary symmetric polynomial of degree equal to i. Since each of the elementary symmetric polynomials is computable by a $\mathrm{poly}(n)$ size depth-3 circuit (by a well-known construction of Ben-Or), if Q has a small circuit, then so does P. A natural question is to understand if a small circuit for P implies that Q must also have a small circuit. Such an upper bound is indeed true in a fairly strong sense as shown in [BJ19, BKR+25] which showed the circuit complexity, formula complexity, or constant-depth circuit complexity of P and Q are polynomially bounded in each other. Theorem 2.3 gives a more fine grained statement of this flavor in the border complexity setting and shows that when Q is homogeneous, the border circuit complexities of P and Q are within sub-linear factors of each other!

The proof of the above statement essentially follows the template of [BJ19]. In [BJ19], the authors consider the polynomial $H(\mathbf{x}, z) = z^n - x_1 z^{n-1} + \cdots + (-1)^n x_n - (-1)^n$. Note that $H(\vec{0}, z) = z^n - 1$ is a square-free polynomial. So, by Newton Iteration, the polynomial $H(\vec{0}, z)$ splits into linear factors $H(\mathbf{x}, z) = (z - A_1(\mathbf{x})) \cdots (z - A_n(\mathbf{x}))$, where $A_i(\mathbf{x})$ are power series roots such that $H(\mathbf{x}, A_i(\mathbf{x})) = 0$. By expanding out this factorization of $H(\mathbf{x}, z)$, we see that

$$\mathbf{Esym}_i(A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})) = \begin{cases} x_i & \text{if } 1 \le i \le n-1, \\ x_n - 1 & \text{if } i = n. \end{cases}$$

Thus, we have

$$\begin{aligned} P(A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})) &= Q(\mathbf{Esym}_1(A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})), \ldots, \mathbf{Esym}_n(A_1(\mathbf{x}), \ldots, A_n(\mathbf{x}))) \\ &= Q(x_1, \ldots, x_{n-1}, x_n - 1). \end{aligned}$$

Let $\widetilde{A}_i(\mathbf{x})$ be the power series approximation up to degree $D+1$ of $A_i(\mathbf{x})$. Then $P(\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})) = Q(x_1, \ldots, x_n - 1) + O(\mathbf{x}^{D+1})$.

We use Newton Iteration to compute all the power series approximations $\widetilde{A}_i(\mathbf{x})$ that are correct up to monomials of degree at least $D = \deg(Q)$. We implement this with a near-linear-size circuit by using univariate multipoint evaluation combined with Newton iteration implemented with

quadratic convergence per iteration (see Theorem 5.6). Then, we feed these approximations as input to a circuit computing $P(x_1, \ldots, x_n)$. Since the root approximations are correct up to monomials of degree $D = \deg(Q)$, we get that the resulting circuit $\widetilde{C}$ computes a polynomial of the form $Q(x_1, \ldots, x_n - 1) + \widetilde{Q}(x_1, \ldots, x_n - 1)$, where $\widetilde{Q}$ has the property that every non-zero monomial in it has degree strictly greater than D in the variables $x_1, x_2, \ldots, (x_n - 1)$. In other words, $\widetilde{Q}$ is in the ideal $\langle x_1, x_2, \ldots, (x_n - 1)\rangle^{D+1}$. Thus, if we could extract homogeneous components of degree at most D from $\widetilde{C}$ with a small blow up in size, we would have a circuit for $Q(x_1, \ldots, x_n - 1)$. Shifting the variable x_n would then give us a circuit for Q. Such a homogeneous component extraction can easily be done with a polynomial in D blow up in size, and this is the route that the authors in [BJ19] take to complete their proof. However, given that we are looking for circuits of near-linear size for Q, we cannot tolerate this $\mathrm{poly}(D)$ blow up in size in our context. At this point, we note if Q happened to be homogeneous in the variables $x_1, \ldots, x_n - 1$, then a circuit that computes Q in the border complexity sense can be extracted from $\widetilde{C}$ with *no* blow up in size! To do this, we take a fresh variable ε and for $i \leq (n-1)$, replace the variable x_i by εx_i and $(x_{n-1} - 1)$ by $\varepsilon(x_n - 1)$.[4] The resulting circuit C' computes a polynomial of the form $(\varepsilon^D Q(x_1, \ldots, x_n - 1) + \varepsilon^{D+1} Q'(x_1, \ldots, x_n - 1, \varepsilon))$ where Q' is a polynomial in $(n+1)$ variables. Dividing C' by ε^D at the output level gives us a circuit that computes $Q(x_1, \ldots, x_n - 1)$ in the border sense. Clearly, the size of C' is same as the size of C! After an appropriate shifting of the variable x_n (that requires a little care), this completes the proof of Theorem 2.3.

For our applications of Theorem 2.3, it happens to be the case that the corresponding instances of the polynomial Q are not really homogeneous! Thankfully, for each of these applications, the instances of the polynomial Q that make an appearance happen to be *weighted* homogeneous, in the sense that for an appropriate choice of weights $w_1, w_2, \ldots, w_n \in \mathbb{N}$, we have $Q(\varepsilon^{w_1} x_1, \varepsilon^{w_2} x_2, \ldots, \varepsilon^{w_n} x_n) = \varepsilon^D Q(x_1, x_2, \ldots, x_n)$ for some parameter D. The upper bound on the border complexity of Q as outlined above continues to hold when Q is weighted homogeneous, thus making it appropriate for application towards the proofs of Theorem 2.1, Theorem 2.2 and Theorem 2.4.

3.3 Resultant of $y + f(x)$ and $g(x)$

We start with a brief sketch of the proof of the upper bound for the resultant (Theorem 2.4). The high level structure of this proof is quite similar to the proof of the results on modular composition and GCD, but the technical details happen to be cleaner and instructive. Recall that the goal is to show that $\mathrm{Res}_x(y + f(x), g(x))$ is in the border of nearly-linear-size circuits that take the coefficients of f and g as inputs.

[4] We can assume without loss of generality that $\widetilde{C}$ takes $x_1, x_2, \ldots, x_{n-1}, (x_n - 1)$ as inputs.

Resultant from roots of $g(x)$: We first make a simple observation that given the roots of $g(x)$, there is a nearly-linear-size circuit for computing $\mathrm{Res}_x(y + f(x), g(x))$. Suppose we are given the roots of $g(x)$, namely $\beta_1, \ldots, \beta_m \in \overline{\mathbb{F}}$. By the Poisson formula, we have

$$\mathrm{Res}_x(y + f(x), g(x)) = \prod_{i=1}^{n} (f(\beta_i) + y).$$

Using the above formula, we build a nearly-linear-size circuit that computes $\mathrm{Res}_x(y + f(x), g(x))$ as follows: using univariate multipoint evaluation (see Lemma 4.3), given $\mathrm{coeff}(f)$ and $\beta_1, \ldots, \beta_m$ as inputs, we compute $f(\beta_1), \ldots, f(\beta_m)$. Then we can compute the product $\prod_{i=1}^{n} (f(\beta_i) + y)$ by applying the FFT algorithm for polynomial multiplication in a divide and conquer manner (see Corollary 4.5). The resulting circuit C is a multi-output circuit that takes $\mathrm{coeff}(f)$ and $\beta_1, \ldots, \beta_n$ as inputs and its i^{th} output gate outputs the coefficient of y^i in $\mathrm{Res}_x(y + f(x), g(x))$.

Can we get an equivalent circuit from C that takes as input $\mathrm{coeff}(g)$ instead of the roots $\beta_1, \ldots, \beta_n$? Note that the roots $\beta_1, \ldots, \beta_n$ are closely related to the coefficients of g, since $g_i = (-1)^{n-i} \mathbf{Esym}_{n-i}(\boldsymbol{\beta})$. Because $\mathrm{Res}_x(y + f(x), g(x))$ is symmetric in $\beta_1, \ldots, \beta_n$, a natural approach would be to somehow invoke Theorem 2.3.

The issue of homogeneity: In order to invoke Theorem 2.3 to go from a circuit on the roots of g to coefficients of g, we first observe that for every i, the coefficient of y^i in $\mathrm{Res}_x(y + f(x), g(x))$, which is a polynomial in the coefficients of f and g, is in fact a weighted homogeneous polynomial in these coefficients where the weights have to be chosen carefully. Moreover, the weights do not depend on the index i and one choice of weights based on the degrees of f and g works for all i. This is formally shown in Lemma 7.3.

Stitching it all together: Given the weighted homogeneity of each of coefficients of $\mathrm{Res}_x(y + f(x), g(x))$, the fact that the resultant is symmetric in the roots of g, the near-linear-size circuit for it that takes the roots of g as inputs, and Theorem 2.3, we combine them in the natural way to obtain a near linear size circuit that computes $\mathrm{Res}_x(y + f(x), g(x))$ in the border sense.

The upper bounds for both modular composition and GCD computation follow the template of the upper bound for the resultant. The main steps in the proofs are analogous to those in the proof of Theorem 2.4 outlined above. We briefly mention these main steps together with pointers to the relevant formal statements.

3.4 Modular composition

The main steps in the proof of Theorem 2.1 are as follows. Recall that the input consists of three univariate polynomials f, g, and h of degree n, given as coefficient vectors, and the goal is to

compute the coefficient vector of the polynomial $f(g(x)) \bmod h(x)$. For simplicity, we start by assuming that the polynomial h has n distinct roots (i.e., that the polynomial h is square-free).

Small circuits starting using roots: As a first step, we observe that if we have access to the n distinct roots $\alpha_1, \ldots, \alpha_n$ of h, then $r(x) := (f(g(x)) \bmod h(x))$ is easy to compute. To see this, note that for every i, since $h(\alpha_i) = 0$, we have that $r(\alpha_i)$ must equal $f(g(\alpha_i))$. Thus, if we have the evaluations of r on the roots, we have sufficient information to uniquely reconstruct r via interpolation (since the degree of r is at most $(n-1)$). We start by using fast univariate multipoint evaluation algorithms (interpreted as a near linear size algebraic circuit) from [BM74] to evaluate g on $\alpha_1, \ldots, \alpha_n$. Then, we once again use such a circuit to evaluate f on $g(\alpha_1), \ldots, g(\alpha_n)$. This gives us n distinct evaluations of the polynomial r that can now be combined via a fast implementation of standard Lagrange interpolation to construct the coefficient vector of r. For formal details, we refer to Lemma 8.2. We also note that the coefficients of r are symmetric in the roots of h.

Weighted homogeneity: As the second step, we show that the coefficients of r are again weighted homogeneous polynomials in the coefficients of f, g, and h for an appropriate choice of weights. This is done in Lemma 8.6 and, together with the fact that the coefficients of r are symmetric in the roots of h, sets us up for invoking Theorem 2.2.

Combining the parts: Finally we combine the nearly-linear-size circuit in the first step above with Theorem 2.3 to obtain Theorem 2.1 for the case when h is square-free. See Theorem 8.9 for details.

Handling non-square-free h: Clearly the above proof relies on the fact that h has n distinct roots, since we need n distinct evaluations of r in order to interpolate it. When h is not square-free and has repeated roots, one possibility might be to consider higher order evaluations of r on the roots, where the order of evaluation depends on the multiplicity of the roots. We work with a simpler alternative here, where we *perturb* the polynomial h to a new polynomial $\hat{h}$ that is square-free. By perturbation, we mean that for a new variable ε, the coefficients of $\hat{h}$ is a function of the coefficients of h and ε such that as ε tends to zero, $\hat{h}$ tends to h. Thus the above circuit for modular composition can now be invoked with inputs f, g, and $\hat{h}$. Finally, we note that as ε tends to zero, the output of this circuit approaches $f(g(x)) \bmod \hat{h}(x)$ in the limit, thereby completing the proof of Theorem 2.1. We refer to Theorem 8.12 for the details.

3.5 GCD computation

We now discuss a high level sketch of the main ideas in the proof of Theorem 2.2. The proof is along the lines of the proof of Theorem 2.1.

Before moving ahead, we recall that unlike modular composition or the resultant, the GCD of two polynomials is not a continuous function of the coefficients of these polynomials, and hence we cannot expect the coefficient vector of the GCD to be computable by an algebraic circuit (even with division gates) that takes the coefficients of these polynomials as inputs. However, once we fix the degree of the GCD and only focus on input polynomials with the promise that their GCD has this specified degree, the coefficients of the GCD do indeed become rational functions of the coefficients of the inputs. And thus, we can expect there to be an algebraic circuit computing the GCD from the coefficients of the input polynomials in this setting. We refer to Lemma 9.1 and the discussion preceding it for more details of this point. For this overview, we assume that we have the coefficient vectors of two polynomials f and g of degree n that are *square-free* (i.e., have no repeated roots) and whose GCD has degree *equal* to $d \leq n$ for some d that is fixed.

GCD from roots of f: We define the polynomial $h(x, y)$ to be equal to $(y - x)g(x)$ and denote the (distinct) roots of f by $\{\alpha_1, \ldots, \alpha_n\}$. Since the GCD of f and g has degree d, a subset of exactly d of the roots of f are also roots of g. Without loss of generality, we will assume that $\{\alpha_{n-d+1}, \ldots, \alpha_n\}$ are the d common roots of f and g. To compute the GCD of f and g, we first compute the polynomial $Q(y) := \prod_{i=1}^{n-d} h(\alpha_i, y)$, which equals $\prod_{i=1}^{n-d}(y - \alpha_i)g(\alpha_i)$. Note that $f(y)/Q(y)$ is the GCD of f and g up to multiplication by the non-zero scalar $\prod_{i=1}^{n-d} g(\alpha_i)$. This is non-zero since the the common roots of f and g are precisely $\{\alpha_{n-d+1}, \ldots, \alpha_n\}$, so each $g(\alpha_i)$ in this product must be non-zero. For this overview, we ignore this issue of multiplication by $\prod_{i=1}^{n-d} g(\alpha_i)$ and focus on the computation of $Q(y)$. Once we have the coefficient vector of $Q(y)$, we can divide $f(y)$ by $Q(y)$ to obtain the GCD.

In order to eventually go from roots to coefficients, we compute Q in a way that allows us to invoke Theorem 2.3, i.e., we need the circuit for Q to be symmetric in $\alpha_1, \ldots, \alpha_n$. To do this, we note that $Q(y)$ is precisely equal to the elementary symmetric polynomial of degree $(n - d)$ in the *multiset* $H = \{h(\alpha_i, y) : i \in [n]\}$, since this multiset has precisely $(n - d)$ non-zero elements, namely $\{h(\alpha_i, y) : i \in [n - d]\}$ and d occurrence of zero.

To make this outline work, we show that given the roots of f and the coefficients of g, we can compute the polynomials in the set H by a circuit of polylogarithmic depth and nearly-linear size. At this point, if we could compute the elementary symmetric polynomials of degree $(n - d)$ on H in near-linear size and polylogarithmic depth, we would be done. We don't know how to show this, but using the additional fact that $\mathbf{Esym}_{n-d}(H)$ is the highest-degree non-zero elementary symmetric polynomial in H, we show that $\mathbf{Esym}_{n-d}(H)$ can be computed in the border of near-linear-size and polylogarithmic-depth circuits. Combined with a division, this gives us a circuit that computes the GCD of f and g in the border complexity setting. The formal details are in Lemma 9.4.

From roots to coefficients: The rest of the proof is along the lines of the analogous steps in Theorem 2.1. We show in Lemma 9.5 that the polynomial Q computed above is weighted

homogeneous in the coefficients of f and g with appropriate weights. We then rely on a *border* version of Theorem 2.3 (see Corollary 6.9) to obtain a near-linear-size polylogarithmic-depth circuit that computes Q in the border sense while taking the coefficients of f and g as input. Finally, we show that this circuit can be used together with the coefficient vectors of f to obtain a circuit with similar size and depth that computes the coefficient vector of the GCD of f and g in the border.

4 Prerequisites

- For a natural number $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.
- We use $\mathbb{F}$ to denote a field and write $\overline{\mathbb{F}}$ for its algebraic closure.
- For a univariate polynomial $f(x) = \sum_{i=0}^{n} f_i x^i \in \mathbb{F}[x]$, we define $\mathrm{coeff}(f) \coloneqq (f_0, \ldots, f_n) \in \mathbb{F}^n$ to be the coefficient vector of f.
- For n variables $x_1, \ldots, x_n$, we use $\mathbf{x}$ to denote the tuple $(x_1, \ldots, x_n)$. If the number of variables in $\mathbf{x}$ is not specified explicitly, it will be made clear from context.
- The *i-th elementary symmetric polynomial*, denoted by $\mathbf{Esym}_i(x_1, \ldots, x_n)$, is defined as

$$\mathbf{Esym}_i(\mathbf{x}) \coloneqq \sum_{\substack{S \subseteq [n] \\ |S|=i}} \prod_{j \in S} x_j.$$

- For a univariate polynomial $f(x) \in \mathbb{F}[x]$, we use the notation $[x^n](f)$ to denote the coefficient of x^n in $f(x)$. When f is a multivariate polynomial, such as $f(x, y)$, we view it as an element of $\mathbb{F}[y][x]$. In this case, $[x^n](f)$ represents the coefficient of x^n, which is itself a polynomial in y.

4.1 Some well-known near-linear-size circuits

Given the coefficients of two polynomials $f(x)$ and $g(x) \in \mathbb{F}[x]$ of degree m and n, respectively, computing the coefficients of their product $f(x) \cdot g(x)$ is a fundamental problem in computer algebra. Naively, polynomial multiplication can be computed in quadratic time. Remarkably, using the fast Fourier transform (FFT), it can be computed in $\widetilde{O}(m+n)$ time. Moreover, the FFT algorithm has parallel complexity of $\mathrm{polylog}(m+n)$.

Lemma 4.1 (Polynomial Multiplication [SS71, CK91])**.** *Let $\mathbb{F}$ be a field. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}$, of size $\widetilde{O}(m+n)$ and depth $\mathrm{polylog}(m+n)$, such that for all polynomials $f(x), g(x) \in \mathbb{F}[x]$ of degree equal to m and n, respectively, we have*

$$C_{m,n}\left(\mathrm{coeff}(f), \mathrm{coeff}(g)\right) = \mathrm{coeff}\left(f \cdot g\right).$$

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Another closely related problem is computing the coefficients of the quotient and remainder of $f(x)$ divided by $g(x)$. Sieveking [Sie72] and Kung [Kun74], using Newton Iteration and the FFT, gave an $\widetilde{O}(m+n)$ time algorithm for inverting power series up to a certain degree. Using this, we get a near-linear-time and $\mathrm{polylog}(m+n)$-depth algorithm for polynomial division with remainder.

Lemma 4.2 (Polynomial Division with Remainder [Sie72, Kun74])**.** *Let $\mathbb{F}$ be a field. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}$, of size $\widetilde{O}(m+n)$ and depth $\mathrm{polylog}(m+n)$, such that for all polynomials $f(x), g(x) \in \mathbb{F}[x]$ of degree equal to m and n, respectively, we have*

$$C_{m,n}\left(\mathrm{coeff}(f), \mathrm{coeff}(g)\right) = \left(\mathrm{coeff}(q), \mathrm{coeff}(r)\right),$$

where $q(x)$ and $r(x)$ are, respectively, the quotient and remainder when $f(x)$ is divided by $g(x)$. Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

The problem of univariate multipoint evaluation asks: Given a polynomial $f(x)$ of degree n and a collection of m points $\alpha_1, \ldots, \alpha_m \in \mathbb{F}$, compute the evaluations $f(\alpha_1), \ldots, f(\alpha_m)$. Borodin and Moenck [BM74] used polynomial division with remainder to give a (remainder-tree-based) nearly-linear-time algorithm for multipoint evaluation.

Lemma 4.3 (Univariate Multipoint Evaluation [BM74])**.** *Let $\mathbb{F}$ be a field. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}$, of size $\widetilde{O}(m+n)$ and depth $\mathrm{polylog}(m+n)$, such that for all polynomials $f(x) \in \mathbb{F}[x]$ of degree equal to n and m evaluation points $(\alpha_1, \ldots, \alpha_m)$, we have*

$$C_{m,n}\left(\mathrm{coeff}(f), (\alpha_1, \ldots, \alpha_m)\right) = \left(f(\alpha_1), \ldots, f(\alpha_m)\right).$$

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

The "inverse" problem to univariate multipoint evaluation is univariate interpolation. Given n tuples $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$, there is a unique polynomial f of degree less that n such that for all $i \in [n]$, the polynomial f satisfies $f(\alpha_i) = \beta_i$. The problem of univariate interpolation requires us to find the coefficients of the interpolating polynomial $f(x)$. Borodin and Moenck [BM74] also gave a remainder-tree-based nearly-linear-time algorithm for univariate interpolation.

Lemma 4.4 (Univariate Interpolation [BM74])**.** *Let $\mathbb{F}$ be a field. For every $n \in \mathbb{N}$, there is a multi-output algebraic circuit C_n, defined over the field $\mathbb{F}$, of size $\widetilde{O}(n)$ and depth $\mathrm{poly}(\log(n))$, such that for n interpolation points $((\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n))$, we have*

$$C_n\left((\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\right) = \mathrm{coeff}(f),$$

where $f(x)$ is the unique interpolating polynomial of degree less than n that satisfies $f(\alpha_i) = \beta_i$ for all $i \in [n]$. Moreover, there is an algorithm that, given n as input, outputs C_n in time $\widetilde{O}(n)$.

Using fast polynomial multiplication (Lemma 4.1) in a divide and conquer fashion, we can compute the coefficients of $(x+\alpha_1)\cdots(x+\alpha_n)$, given $(\alpha_1,\ldots,\alpha_n)$, in $\widetilde{O}(n)$ time and polylog(n) depth.

Corollary 4.5 (Multiplying linear forms)**.** *Let $\mathbb{F}$ be a field. For every $n \in \mathbb{N}$, there is a multi-output algebraic circuit C_n, defined over the field $\mathbb{F}$, of size $\widetilde{O}(n)$ and depth* polylog(n)*, such that for n points $(\alpha_1,\ldots,\alpha_n)$, we have*

$$C_n(\alpha_1,\ldots,\alpha_n) = \text{coeff}(f),$$

where $f(x) = (x+\alpha_1)\cdots(x+\alpha_n)$. Moreover, there is an algorithm that, given n as input, outputs C_n in time $\widetilde{O}(n)$.

Proof. Suppose the circuit C_n has size $s(n)$ and depth $\Delta(n)$. Let $g(x) = (x+\alpha_1)\cdots(x+\alpha_{\lfloor n/2\rfloor})$ and $h(x) = (x+\alpha_{\lfloor n/2\rfloor+1})\cdots(x+\alpha_n)$. In order to build C_n, we first compute coeff(g) and coeff(h) and then use Lemma 4.1 to compute the coefficients of the product $f(x) = g(x)\cdot h(x)$. Recursively, we use $C_{\lfloor n/2\rfloor}$ to compute coeff(g) and $C_{\lceil n/2\rceil}$ to compute coeff(h). Thus, we have the following size and depth bounds for C_n,

$$\begin{aligned} s(n) &\le s(\lfloor n/2\rfloor) + s(\lceil n/2\rceil) + \widetilde{O}(n) \\ \Delta(n) &\le \Delta(\lceil n/2\rceil) + \text{polylog}(n). \end{aligned}$$

Trivially, we have that $s(1) = O(1)$ and $\Delta(1) = O(1)$. Thus, solving the above recurrence, we get the required size and depth bounds for C_n. Also, the above construction gives an algorithm that, given n as input, outputs C_n in time $\widetilde{O}(n)$. □

The coefficients of $(x+\alpha_1)\cdots(x+\alpha_n)$ are the evaluations of the elementary symmetric polynomials $\{\mathbf{Esym}_i(\boldsymbol{\alpha}) : i \in [n]\}$. Thus, Corollary 4.5 outputs all the elementary symmetric polynomials in $\alpha_1,\ldots,\alpha_n$. Since we will use this observation often, we formally state it as a corollary below.

Corollary 4.6 (Computing elementary symmetric polynomials)**.** *Let $\mathbb{F}$ be a field. For every $n \in \mathbb{N}$, there is a multi-output algebraic circuit C_n, defined over the field $\mathbb{F}$, of size $\widetilde{O}(n)$ and depth* polylog(n)*, such that for n points $(\alpha_1,\ldots,\alpha_n)$, we have*

$$C_n(\alpha_1,\ldots,\alpha_n) = \left(\mathbf{Esym}_1(\boldsymbol{\alpha}),\ldots,\mathbf{Esym}_n(\boldsymbol{\alpha})\right).$$

Moreover, there is an algorithm that, given n as input, outputs C_n in time $\widetilde{O}(n)$.

Proof. We give $(\alpha_1,\ldots,\alpha_n)$ as input to circuit C from Corollary 4.5 and this gives the required output. The size and depth bounds for C follow from Corollary 4.5. □

4.2 Weighted homogeneous polynomials

Weighted homogeneous polynomials will play a crucial role in the following sections. We first define the standard notion of homogeneous polynomials.

Definition 4.7. *Let t be a formal variable. A polynomial $Q(x_1, \ldots, x_n) \in \mathbb{F}[\mathbf{x}]$ is said to be* homogeneous of degree D *if*

$$Q(tx_1, tx_2, \ldots, tx_n) = t^D Q(x_1, \ldots, x_n). \qquad \Diamond$$

The above definition can be generalized to weighted homogeneous polynomials as follows.

Definition 4.8. *Let t be a formal variable. For weights $d_1, d_2, \ldots, d_n \in \mathbb{Z}_{\geq 0}$, a polynomial $Q(x_1, \ldots, x_n) \in \mathbb{F}[\mathbf{x}]$ is said to be a $(d_1, d_2, \ldots, d_n)$-*homogeneous polynomial *if there is a natural number $D \in \mathbb{N}$ such that*

$$Q(t^{d_1}x_1, t^{d_2}x_2, \ldots, t^{d_n}x_n) = t^D Q(x_1, \ldots, x_n).$$

We say that D is the weighted degree *of Q.* $\Diamond$

Let $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i \in \mathbb{F}[x]$ be a monic polynomial with roots $\alpha_1, \ldots, \alpha_n \in \overline{\mathbb{F}}$. We will often work with polynomial expressions over the roots $P(\alpha_1, \ldots, \alpha_n)$ and over the coefficients $Q(f_0, f_1, \ldots, f_{n-1})$ such that $P(\alpha_1, \ldots, \alpha_n) = Q(f_0, \ldots, f_{n-1})$. When P is a homogeneous polynomial, we observe below that Q is weighted homogeneous.

Lemma 4.9. *Let $P(x_1, \ldots, x_n)$ and $Q(x_1, \ldots, x_n) \in \mathbb{F}[\mathbf{x}]$ be polynomials such that for all monic polynomials $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i \in \mathbb{F}[x]$ of degree n, we have*

$$P(\alpha_1, \ldots, \alpha_n) = Q(f_0, f_1, \ldots, f_{n-1}),$$

where $\alpha_1, \ldots, \alpha_n \in \overline{\mathbb{F}}$ are the roots of $f(x)$. If $P(\mathbf{x})$ is a homogeneous polynomial of degree D, then $Q(\mathbf{x})$ is a $(n, n-1, \ldots, 1)$-homogeneous polynomial of weighted degree D.

Proof. Let t be a formal variable. Consider the polynomial $F(x) = x^n + \sum_{i=0}^{n-1} t^{n-i} f_i x^i$. For each $i \in \{0, 1, \ldots, n-1\}$, we have $F(t\alpha_i) = t^n f(\alpha_i) = 0$. Thus, $t\alpha_1, \ldots, t\alpha_n$ are the roots of $F(x)$. By the assumed relationship between P and Q, we have

$$\begin{aligned} Q(t^n f_0, t^{n-1} f_1, \ldots, t f_{n-1}) &= P(t\alpha_1, \ldots, t\alpha_n) \\ &= t^D P(\alpha_1, \ldots, \alpha_n) \qquad \text{(Since } P \text{ is degree } D \text{ homogeneous)} \\ &= t^D Q(f_0, f_1, \ldots, f_{n-1}). \end{aligned}$$

Thus, $Q(\mathbf{x})$ is a $(n, n-1, \ldots, 1)$-homogeneous polynomial of weighted degree D. $\square$

Lemma 4.9 can be generalized to multiple univariate polynomials. For the ease of exposition and notational convenience, we extend Lemma 4.9 for two polynomials below.

Lemma 4.10. *Let $P(x_1, \ldots, x_n, y_1, \ldots, y_m)$ and $Q(x_1, \ldots, x_n, y_1, \ldots, y_m) \in \mathbb{F}[\mathbf{x}, \mathbf{y}]$ be polynomials such that for all monic polynomials $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i \in \mathbb{F}[x]$ of degree n and $g(x) = x^m + \sum_{i=0}^{m-1} g_i x^i \in \mathbb{F}[x]$ of degree m, we have*

$$P(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) = Q(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}),$$

where $\alpha_1, \ldots, \alpha_n \in \overline{\mathbb{F}}$ and $\beta_1, \ldots, \beta_m \in \overline{\mathbb{F}}$ are the roots of $f(x)$ and $g(x)$, respectively. If $P(\mathbf{x}, \mathbf{y})$ is a homogeneous polynomial of degree D, then $Q(\mathbf{x}, \mathbf{y})$ is a $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous polynomial of weighted degree D.

Proof. Let t be a formal variable. As in the proof of the Lemma 4.9, consider the polynomials $F(x) = x^n + \sum_{i=0}^{n-1} t^{n-i} f_i x^i$ and $G(x) = x^m + \sum_{i=0}^{m-1} t^{m-i} g_i x^i$. For each $i \in [n]$, we have $F(t\alpha_i) = t^n f(\alpha_i) = 0$, and for each $j \in [m]$, we have $G(t\beta_j) = t^m g(\beta_j) = 0$. Thus, $t\alpha_1, \ldots, t\alpha_n$ are the roots of $F(x)$ and $t\beta_1, \ldots, t\beta_m$ are the roots of $G(x)$. By the assumed relationship between P and Q, we have

$$\begin{aligned}
&Q(t^n f_0, t^{n-1} f_1, \ldots, t f_{n-1}, t^m g_0, t^{m-1} g_1, \ldots, t g_{m-1}) \\
&= P(t\alpha_1, \ldots, t\alpha_n, t\beta_1, \ldots, t\beta_n) \\
&= t^D P(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n) && \text{(Since } P \text{ is degree } D \text{ homogeneous)} \\
&= t^D Q(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}).
\end{aligned}$$

Thus $Q(\mathbf{x}, \mathbf{y})$ is a $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous polynomial of weighted degree D. □

4.3 Newton Iteration

In this subsection, we collect some standard facts related to Newton iteration. These statements are folklore and we refer to [DSS22, Section 3] for their proofs.

In both statements, we start with a monic polynomial $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ such that $H(\vec{0}, y)$ is square-free. The first shows that $H(\mathbf{x}, y)$ factors completely into linear terms if we allow the factors to be power series in the $\mathbf{x}$ variables.

Theorem 4.11 (Factorization into power series (see, e.g., [DSS22, Section 3]))**.** *Let $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ be a monic polynomial such that $H(\vec{0}, y)$ is square-free. Suppose $H(\vec{0}, y)$ factors into linear terms as*

$$H(\vec{0}, y) = \prod_{i=1}^{n} (y - \beta_i),$$

where $\beta_1, \ldots, \beta_n \in \overline{\mathbb{F}}$ are distinct field elements. Then there exist unique power series $A_1(\mathbf{x}), \ldots, A_n(\mathbf{x}) \in$

$\overline{\mathbb{F}}[\![\mathbf{x}]\!]$ *such that* $A_i(\vec{0}) = \beta_i$ *and* $P(\mathbf{x}, A_i(\mathbf{x})) = 0$ *for all* $i \in [n]$*. In other words, the polynomial* $H(\mathbf{x}, y)$ *factors as*

$$H(\mathbf{x}, y) = \prod_{i=1}^{n} (y - A_i(\mathbf{x})),$$

in $\overline{\mathbb{F}}[\![\mathbf{x}]\!]$*. Moreover, if* $\beta_1, \ldots, \beta_n$ *lie in the base field* $\mathbb{F}$*, then the coefficients of* $A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})$ *also lie in* $\mathbb{F}$*.*

The second result shows that the coefficients of the power series $A_i(\mathbf{x})$ in the factorization of $H(\mathbf{x}, y)$ can be computed efficiently if we are given $H(\mathbf{x}, y)$ and the factorization of $H(\vec{0}, y)$.

Theorem 4.12 (Newton iteration with quadratic convergence [vzGG13, Lemmas 9.21 and 9.27])**.** *Let* $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ *be a monic polynomial such that* $H(\vec{0}, y)$ *is square-free. Let* $A_1(\mathbf{x}), \ldots, A_n(\mathbf{x}) \in \overline{\mathbb{F}}[\![\mathbf{x}]\!]$ *be power series such that*

$$H(\mathbf{x}, y) = \prod_{i=1}^{n} (y - A_i(\mathbf{x})).$$

For each $i \in [n]$*, consider the sequence of rational functions* $(\varphi_i^{(j)}(\mathbf{x}))_{j \in \mathbb{N}}$ *given by*

$$\varphi_i^{(0)} := A_i(\vec{0})$$
$$\varphi_i^{(j+1)} := \varphi_i^{(j)} - \frac{H(\mathbf{x}, \varphi_i^{(j)})}{\partial_y H(\mathbf{x}, \varphi_i^{(j)})}.$$

Then for all $j \in \mathbb{N}$*, the rational function* $\varphi_i^{(j)}(\mathbf{x})$ *can be expanded as a power series around* $\vec{0}$ *and this power series is an approximation to* $A_i(\mathbf{x})$ *in the sense that*

$$\varphi_i^{(j)}(\mathbf{x}) = A_i(\mathbf{x}) \bmod \langle \mathbf{x} \rangle^{2^j}.$$

5 Fast & parallel Newton Iteration

Newton iteration is a standard way to obtain approximations to power series roots (in y) of a polynomial $H(\mathbf{x}, y)$. In this section, we will observe that Newton iteration yields a shallow, near-linear-size circuit that computes approximations to the power series roots of $H(\mathbf{x}, y)$. We first define the notion of degree-D approximations of y-roots of $H(\mathbf{x}, y)$.

Definition 5.1. *Let* $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ *be a polynomial, and suppose there are power series* $A_1, \ldots, A_n \in \overline{\mathbb{F}}[\![\mathbf{x}]\!]$ *such that* $H(\mathbf{x}, y)$ *factors as*

$$H(\mathbf{x}, y) = \prod_{i=1}^{n} (y - A_i(\mathbf{x})).$$

We say that a tuple of polynomials $(\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})) \in \mathbb{F}[\mathbf{x}]^n$ *are* degree-D approximations of the y-roots of $H(\mathbf{x}, y)$ *if for all* $i \in [n]$*, we have*

$$\widetilde{A}_i(\mathbf{x}) = A_i(\mathbf{x}) \bmod \langle \mathbf{x} \rangle^{D+1}. \qquad \lozenge$$

As a first step, we show that Newton iteration yields a small, shallow circuit that computes rational functions whose power series expansions correspond to approximate y-roots of a given polynomial $H(\mathbf{x}, y)$.

Lemma 5.2. *Let* $\mathbb{F}$ *be a field. For every* $n, D \in \mathbb{N}$*, there is a multi-output algebraic circuit* $C_{n,D}$ *defined over the field* $\mathbb{F}$ *such that the following is true. The circuit* $C_{n,D}$ *has size* $\widetilde{O}(n \log D)$*, depth* $\operatorname{polylog}(n, D)$*, and for all polynomials* $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ *with* y*-degree* n *such that* $H(\vec{0}, y)$ *is square-free, we have*

$$C_{n,D}(\operatorname{coeff}_y(H), (\beta_1, \ldots, \beta_n)) = (\varphi_1(\mathbf{x}), \ldots, \varphi_n(\mathbf{x})),$$

where $\operatorname{coeff}_y(H)$ *are the* y*-coefficients of* $H(\mathbf{x}, y)$*,* $\beta_1, \ldots, \beta_n \in \overline{\mathbb{F}}$ *are the roots of* $H(\vec{0}, y)$*, and* $\varphi_1(\mathbf{x}), \ldots, \varphi_n(\mathbf{x}) \in \overline{\mathbb{F}}(\mathbf{x})$ *are rational functions that admit a power series expansion around* $\vec{0}$ *and are degree-*D *approximations of the* y*-roots of* $H(\mathbf{x}, y)$*. Moreover, there is an algorithm that, given* n *and* D *as inputs, outputs the circuit* $C_{n,D}$ *in time* $\widetilde{O}(n \log D)$*.*

Remark 5.3. *If the roots* $\beta_1, \ldots, \beta_n$ *of* $H(\vec{0}, y)$ *above are in* $\mathbb{F}$*, then the approximate roots* $\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})$ *are in* $\mathbb{F}[\mathbf{x}]$*.* $\lozenge$

Proof of Lemma 5.2. Since $H(\vec{0}, y)$ is square-free, we will use Newton Iteration (Theorem 4.12) to get a sequence of rational functions whose power series expansion around $\vec{0}$ are approximations to the roots of $H(\mathbf{x}, y)$. Let $A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})$ be the power series roots of $H(\mathbf{x}, y)$. The rational functions approximating the i^{th} root $A_i(\mathbf{x})$ are given by

$$\begin{aligned} \varphi_i^{(0)}(\mathbf{x}) &\coloneqq \beta_i \\ \varphi_i^{(j+1)}(\mathbf{x}) &\coloneqq \varphi_i^{(j)}(\mathbf{x}) - \frac{H(\mathbf{x}, \varphi_i^{(j)}(\mathbf{x}))}{\partial_y H(\mathbf{x}, \varphi_i^{(j)}(\mathbf{x}))}. \end{aligned}$$

These rational functions can be expanded as power series around $\vec{0}$, and these power series converge to the power series root $A_i(\mathbf{x})$ in the sense that

$$\varphi_i^{(j)}(\mathbf{x}) = A_i(\mathbf{x}) \bmod \langle \mathbf{x} \rangle^{2^j}.$$

To see why $\varphi_i^{(j)}$ admits a power series expansion around $\vec{0}$, first note that $\varphi_i^{(0)}$ is a constant, and so trivially admits a power series expansion. When $j \geq 1$, we may assume by induction that $\varphi_i^{(j-1)}$

can be expanded as a power series around $\vec{0}$. We observe that

$$\partial_y H(\vec{0}, \varphi_i^{(j-1)}(\vec{0})) = \partial_y H(\vec{0}, \beta_i) \neq 0,$$

since $H(\vec{0}, y)$ is square-free and β_i is a root of $H(\vec{0}, y)$. This implies that $1/\partial_y H(\mathbf{x}, \varphi_i^{(j-1)}(\mathbf{x}))$ can be expanded as a power series around $\vec{0}$, so it follows that $\varphi_i^{(j)}$ also admits a power series expansion at $\vec{0}$.

Let $k := \lceil \log(D+1) \rceil$. Note that

$$\varphi_i^{(k)}(\mathbf{x}) = A_i(\mathbf{x}) \bmod \langle \mathbf{x} \rangle^{D+1},$$

so $\varphi_i^{(k)}(\mathbf{x})$ will provide a sufficiently good approximation of $A_i(\mathbf{x})$. We will construct a circuit C that iteratively computes the rational functions $\varphi_i^{(j)}$ for $j \in \{0, 1, \ldots, k\}$. The function $\varphi_i^{(j+1)}$ can be expressed as

$$\varphi_i^{(j+1)} = \frac{N(\varphi_i^{(j)})}{D(\varphi_i^{(j)})}, \tag{5.4}$$

where $N(y) := y \cdot \partial_y H(\mathbf{x}, y) - H(\mathbf{x}, y)$ and $D(y) := \partial_y H(\mathbf{x}, y)$ are polynomials in $\mathbb{F}[\mathbf{x}][y]$. Since we are given $\mathrm{coeff}_y(H)$ as input, the coefficients of $N(y)$ and $D(y)$ are computable by a circuit of size $O(n)$.

In order to construct C, we first construct a circuit $C^{(j)}$ such that

$$C^{(j)}(\mathrm{coeff}(N), \mathrm{coeff}(D), (\varphi_1^{(j)}, \ldots, \varphi_n^{(j)})) = (\varphi_1^{(j+1)}, \ldots, \varphi_n^{(j+1)}).$$

To do this, we use the multipoint evaluation circuit from Lemma 4.3 to compute $(N(\varphi_1^{(j)}), \ldots, N(\varphi_n^{(j)}))$ and $(D(\varphi_1^{(j)}), \ldots, D(\varphi_n^{(j)}))$. Then we use Eq. (5.4) to compute $(\varphi_1^{(j+1)}, \ldots, \varphi_n^{(j+1)})$ using division gates. From Lemma 4.3, the resulting circuit $C^{(j)}$ has size $\widetilde{O}(n)$ and depth $\mathrm{poly}(\log n)$. Moreover, from Lemma 4.3, there is an algorithm that constructs the circuit $C^{(j)}$ in $\widetilde{O}(n)$ time.

Using the circuits $C^{(0)}, \ldots, C^{(k-1)}$, we build the circuit C by feeding the output of $C^{(j-1)}$ as input to $C^{(j)}$, for each $j \in [k-1]$. Thus, the circuit C computes the rational functions $(\varphi_1^{(k)}, \ldots, \varphi_n^{(k)})$. Since $k = \lceil \log(D) \rceil$, the circuit C has size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$. Also, the above construction gives an algorithm that, given n and D as inputs, outputs the circuit C in $\widetilde{O}(n \log D)$ time. □

For our later applications, it will be convenient to have a small, shallow circuit that outputs *polynomials*, not rational functions, that are approximate roots of $H(\mathbf{x}, y)$. We do this by applying Strassen's division elimination [Str73] to the circuit produced by Lemma 5.2. We first recall Strassen's division elimination procedure, noting that it can be carried out in a uniform manner.

Lemma 5.5 ([Str73]). *Let $\mathbb{F}$ be a field. For every multi-output algebraic circuit C of size s and depth Δ that computes a family of rational functions $\varphi_1(\mathbf{x}), \ldots, \varphi_m(\mathbf{x}) \in \mathbb{F}(\mathbf{x})$, there is a multi-output division-free algebraic circuit C' of size $O(s)$ and depth $O(\Delta)$ that computes polynomials $f_1, \ldots, f_m, g_1, \ldots, g_m \in \mathbb{F}[\mathbf{x}]$ such that*

$$\varphi_i(\mathbf{x}) = \frac{f_i(\mathbf{x})}{g_i(\mathbf{x})}$$

for all $i \in [m]$. Furthermore, there is an algorithm that, when given the circuit C as input, outputs the circuit C' in $\widetilde{O}(s)$ time.

Proof. For each gate v of C, we create two new gates labeled (v, num) and (v, den). These gates have the intended meaning that if v computes a rational function $\varphi(\mathbf{x})$, then (v, num) and (v, den) compute polynomials $f(\mathbf{x})$ and $g(\mathbf{x})$, respectively, so that $\varphi(\mathbf{x}) = f(\mathbf{x})/g(\mathbf{x})$. If v is an addition gate with children u and w, then we wire (v, num) and (v, den) as

$$\begin{aligned}(v, \mathrm{num}) &= (u, \mathrm{num}) \times (w, \mathrm{den}) + (u, \mathrm{den}) \times (w, \mathrm{num}) \\ (v, \mathrm{den}) &= (u, \mathrm{den}) \times (w, \mathrm{den}).\end{aligned}$$

Similarly, if v is a multiplication gate with children u and w, we wire (v, num) and (v, den) as

$$\begin{aligned}(v, \mathrm{num}) &= (u, \mathrm{num}) \times (w, \mathrm{num}) \\ (v, \mathrm{den}) &= (u, \mathrm{den}) \times (w, \mathrm{den}).\end{aligned}$$

Finally, if v is a division gate with children u and w, where $v = u/w$, we set

$$\begin{aligned}(v, \mathrm{num}) &= (u, \mathrm{num}) \times (w, \mathrm{den}) \\ (v, \mathrm{den}) &= (u, \mathrm{den}) \times (w, \mathrm{num}).\end{aligned}$$

It is straightforward to check that the gates of this new circuit compute correctly. Moreover, this re-wiring only increases the size and depth of the circuit by a factor of two and can be performed in $\widetilde{O}(s)$ time. □

By combining Lemmas 5.2 and 5.5, we obtain a small, shallow circuit that computes approximate roots of a given polynomial $H(\mathbf{x}, y)$.

Theorem 5.6. *Let $\mathbb{F}$ be a field. For every $n, D \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{n,D}$ defined over the field $\mathbb{F}$ such that the following is true. The circuit $C_{n,D}$ has size $\widetilde{O}(n \log D)$, depth $\mathrm{polylog}(n, D)$, and for all polynomials $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ with y-degree n such that $H(\vec{0}, y)$ is square-free, we have*

$$C_{n,D}(\mathrm{coeff}_y(H), \mathrm{coeff}_y(H)|_{\mathbf{x}=\vec{0}}, (\beta_1, \ldots, \beta_n)) = (\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})),$$

where $\mathrm{coeff}_y(H)$ *are the* y*-coefficients of* $H(\mathbf{x}, y)$, $\mathrm{coeff}_y(H)|_{\mathbf{x}=\vec{0}}$ *are the coefficients of* $H(\vec{0}, y)$, $\beta_1, \ldots, \beta_n \in \overline{\mathbb{F}}$ *are the roots of* $H(\vec{0}, y)$, *and* $\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x}) \in \overline{\mathbb{F}}[\mathbf{x}]$ *are degree-*D *approximations of the* y*-roots of* $H(\mathbf{x}, y)$. *Moreover, there is an algorithm that, given* n *and* D *as inputs, outputs the circuit* $C_{n,D}$ *in time* $\widetilde{O}(n \log D)$.

Proof. By invoking Lemmas 5.2 and 5.5, we obtain a circuit C' of size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$ that computes polynomials $f_1, \ldots, f_n, g_1, \ldots, g_n \in \mathbb{F}[\mathbf{x}]$ such that the rational function

$$\varphi_i(\mathbf{x}) := \frac{f_i(\mathbf{x})}{g_i(\mathbf{x})}$$

has a power series expansion around $\vec{0}$ and is a degree-D approximation to the i-th power series root of $H(\mathbf{x}, y)$. We will obtain the polynomial approximation to the power series roots of $H(\mathbf{x}, y)$ by expanding φ_i as a power series to sufficiently high accuracy. To do this, we need to verify that $g_i(\mathbf{x})$ is invertible in the ring of formal power series, which amounts to showing that $g_i(\vec{0}) \neq 0$.[5]

Recall that the Newton iteration circuit constructed by Lemma 5.2 proceeds by constructing a sequence of rational functions $(\varphi_i^{(0)}, \varphi_i^{(1)}, \ldots, \varphi_i^{(k)})$, of which φ_i is the final element. Let $f_{i,j}, g_{i,j} \in \mathbb{F}[\mathbf{x}]$ be the polynomials computed as the numerator and denominator of $\varphi_i^{(j)}$, respectively, by applying the division elimination of Lemma 5.5 to the Newton iteration circuit. We will prove by induction on j that $g_{i,j}(\vec{0}) \neq 0$. Taking $j = k$ proves that $g_i(\vec{0}) = g_{i,k}(\vec{0}) \neq 0$ as desired.

For $j = 0$, the claim is obvious: we have $g_{i,0}(\mathbf{x}) = 1$, since no divisions have occurred in the circuit. For $j > 0$, we inspect the divisions occurring in one step of Newton iteration. Write $H(\mathbf{x}, y) = \sum_{i=0}^{n} H_i y^i$. One step of Newton iteration assigns

$$\varphi_i^{(j)} = \varphi_i^{(j-1)} - \frac{H(\mathbf{x}, \varphi_i^{(j-1)})}{\partial_y H(\mathbf{x}, \varphi_i^{(j-1)})}.$$

Expanding this over a common denominator without canceling terms, we see that this writes $\varphi_i^{(j)}$ as

$$\varphi_i^{(j)}(\mathbf{x}) = \frac{\sum_{k=0}^{n} k H_k f_{i,j-1}(\mathbf{x})^k g_{i,j-1}(\mathbf{x})^{n-k} - \sum_{k=0}^{n} H_k f_{i,j-1}(\mathbf{x})^k g_{i,j-1}(\mathbf{x})^{n-k}}{\sum_{k=0}^{n} k H_k f_{i,j-1}(\mathbf{x})^{k-1} g_{i,j-1}(\mathbf{x})^{n+1-k}}.$$

Because we did not cancel any intermediate terms, this implies that the division elimination

[5] This is not immediate from the fact that $\varphi_i(\mathbf{x})$ itself has a power series expansion around $\vec{0}$. It is *a priori* possible that f_i and g_i share a common factor h_i that is zero at $\vec{0}$, in which case g_i is not invertible as a power series despite the fact that $\varphi_i = f_i / g_i$ can be expanded around $\vec{0}$.

procedure assigns

$$\begin{aligned} g_{i,j}(\mathbf{x}) &:= \sum_{k=0}^{n} kH_k f_{i,j-1}(\mathbf{x})^{k-1} g_{i,j-1}(\mathbf{x})^{n+1-k} \\ &= g_{i,j-1}(\mathbf{x})^n \sum_{k=0}^{n} kH_k \frac{f_{i,j-1}(\mathbf{x})^{k-1}}{g_{i,j-1}(\mathbf{x})^{k-1}} \\ &= g_{i,j-1}(\mathbf{x})^n \partial_y H(\mathbf{x}, \varphi_i^{(j-1)}(\mathbf{x})). \end{aligned}$$

By induction on j, we have $g_{i,j-1}(\vec{0})^n \neq 0$. Since $H(\vec{0}, y)$ is squarefree and $\varphi_i^{(j-1)}(\vec{0}) = \beta_i$ is a root of $H(\vec{0}, y)$, we have

$$\partial_y H(\vec{0}, \varphi_i^{(j-1)}(\vec{0})) = \partial_y H(\vec{0}, \beta_i) \neq 0.$$

Thus, we have $g_{i,j}(\vec{0}) \neq 0$ as claimed.

Now that we have established $g_i(\vec{0}) \neq 0$, we proceed to expand $\varphi_i(\mathbf{x})$ as a power series around $\vec{0}$. Let $\gamma_i := g_i(\vec{0})$ and let $\hat{g}_i(\mathbf{x}) := g_i(\mathbf{x}) - \gamma_i$. Then we have the equality of power series

$$\frac{1}{g_i(\mathbf{x})} = \frac{1}{\gamma_i + \hat{g}_i(\mathbf{x})} = \frac{1}{\gamma_i} \cdot \frac{1}{1 + \frac{\hat{g}_i(\mathbf{x})}{\gamma_i}} = \frac{1}{\gamma_i} \sum_{j=0}^{\infty} \left(\frac{-\hat{g}_i(\mathbf{x})}{\gamma_i} \right)^j.$$

Because $\hat{g}_i(\mathbf{x})$ has no constant term, every monomial of $\hat{g}_i(\mathbf{x})^i$ has degree at least i. This implies

$$\varphi_i(\mathbf{x}) = \frac{f_i(\mathbf{x})}{g_i(\mathbf{x})} = \frac{f_i(\mathbf{x})}{\gamma_i} \sum_{j=0}^{\infty} \left(\frac{-\hat{g}_i(\mathbf{x})}{\gamma_i} \right)^j = \frac{f_i(\mathbf{x})}{\gamma_i} \sum_{j=0}^{D} \left(\frac{-\hat{g}_i(\mathbf{x})}{\gamma_i} \right)^j \bmod \langle \mathbf{x} \rangle^{D+1}.$$

Define

$$\widetilde{A}_i(\mathbf{x}) := \frac{f_i(\mathbf{x})}{\gamma_i} \sum_{j=0}^{D} \left(\frac{-\hat{g}_i(\mathbf{x})}{\gamma_i} \right)^j.$$

Then we have

$$\widetilde{A}_i(\mathbf{x}) = \varphi_i(\mathbf{x}) = A_i(\mathbf{x}) \bmod \langle \mathbf{x} \rangle^{D+1},$$

so $\widetilde{A}_i(\mathbf{x})$ is a degree-D approximation of the power series $A_i(\mathbf{x})$. Our goal will be to modify the circuit to output $\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})$.

We already have a circuit of size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$ that computes the polynomials $f_1, \ldots, f_n, g_1, \ldots, g_n$. By making a second copy of this circuit, setting $\mathbf{x} = \vec{0}$, and replacing $\mathrm{coeff}_y(H)$ with $\mathrm{coeff}_y(H)|_{\mathbf{x}=\vec{0}}$, we obtain a circuit of the same size and depth that computes the scalars $\gamma_1, \ldots, \gamma_n$. This allows us to compute $f_i(\mathbf{x})/\gamma_i$ and $-\hat{g}_i(\mathbf{x})/\gamma_i$ in the same size and depth.

To compute $\widetilde{A}_i(\mathbf{x})$, we use the fact that

$$1 + y + y^2 + y^3 + \cdots + y^D = (1+y)(1+y^2)(1+y^4)\cdots(1+y^{\frac{D+1}{2}}),$$

where for notational convenience we assume, without loss of generality, that $D+1$ is a power of 2. (We may ensure that $D+1$ is a power of 2 by at most doubling D, which will not affect the final bounds on the size and depth of our circuit.) This lets us write

$$\widetilde{A}_i(\mathbf{x}) = \frac{f_i(\mathbf{x})}{\gamma_i} \prod_{j=0}^{\log_2(D+1)-1} \left(\frac{-\hat{g}_i(\mathbf{x})}{\gamma_i}\right)^{2^j}.$$

We compute all needed powers of $-\hat{g}_i(\mathbf{x})/\gamma_i$ using repeated squaring, which costs $O(\log D)$ size and depth. Overall, this allows us to compute each $\widetilde{A}_i(\mathbf{x})$ with $O(\log D)$ additional size and depth. In total, this results in a circuit of size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$ that computes $\widetilde{A}_1(\mathbf{x}), \ldots, \widetilde{A}_n(\mathbf{x})$. This last part of the circuit to compute the expansion $\widetilde{A}_i$ from the rational function f_i/g_i can also be printed in $\widetilde{O}(n \log D)$ time, as desired. □

6 Border Complexity of Symmetric Polynomials

A polynomial $P(x_1, \ldots, x_n)$ is said to be *symmetric* if it is invariant under re-ordering of variables. The elementary symmetric polynomials $\mathbf{Esym}_i(x_1, \ldots, x_n)$ are closely related to symmetric polynomials in the following sense: Every symmetric polynomial can be written as a composition of some polynomial with the elementary symmetric polynomials. Moreover, there is a unique way to write P as a polynomial function of the elementary symmetric polynomials. This is the fundamental theorem of symmetric polynomials, which we now quote.

Theorem 6.1 (Fundamental Theorem of Symmetric Polynomials)**.** *Let $P(x_1, \ldots, x_n)$ be a symmetric polynomial of degree d. Then there is a unique polynomial $Q(y_1, \ldots, y_n)$ of degree at most d such that*

$$P(x_1, \ldots, x_n) = Q(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x})).$$

Bläser and Jindal [BJ19] extended the above theorem to the computational setting. Suppose $P(x_1, \ldots, x_n)$ is computable by a small algebraic circuit. What can we say about the circuit computing $Q(y_1, \ldots, y_n)$? Bläser and Jindal showed that $Q(y_1, \ldots, y_n)$ can also be computed by a small circuit. In particular, if P can be computed by a circuit of size s, then Q can be computed by a circuit of size $\mathrm{poly}(s)$. Bhattacharjee et al. [BKR+25] later extended this result to both formula complexity and constant-depth circuit complexity.

Motivated by this and our later applications, we ask the following question: If $P(x_1, \ldots, x_n)$ is computable by a circuit of size s, then is $Q(y_1, \ldots, y_n)$ computable by circuits of size $\widetilde{O}(s)$? Note

that here we are asking for a very fine-grained version of the result proved by Bläser and Jindal. Although we are not able to answer this question, we show a weaker statement of similar flavor below. It is weaker in two aspects: we assume that $Q(y_1, \ldots, y_n)$ is a (weighted) homogeneous polynomial, and we show that Q is in the border of circuits of size $\widetilde{O}(s)$. These restrictions suffice for showing near-linear-size border upper bounds for our applications.

Theorem 6.2 (Border Complexity of Symmetric polynomials)**.** *Let $\mathbb{F}$ be a field of size at least n and ε be a formal variable. Let $P(x_1, \ldots, x_n) \in \mathbb{F}[\mathbf{x}]$ be a symmetric polynomial that is computable by an algebraic circuit C of size s and depth Δ, and let $Q(z_1, \ldots, z_n) \in \mathbb{F}[\mathbf{z}]$ be the unique polynomial such that $P(\mathbf{x}) = Q(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x}))$.*

Suppose $Q(\mathbf{z})$ is a homogeneous polynomial of degree D. Then there is an algebraic circuit C', defined over the field $\mathbb{F}(\varepsilon)$, of size $\widetilde{O}(s + n \log D)$ and depth $\Delta + \mathrm{polylog}(n, D)$, that outputs $Q(\mathbf{z}) + O(\varepsilon)$. Moreover, there is an algorithm that, given D and the circuit C as input, outputs the circuit C' in $\widetilde{O}(s + n \log D)$ time.

We will typically be interested in the regime of parameters where $D \leq \mathrm{poly}(n)$. In this case, the above theorem gives us a circuit of size $\widetilde{O}(s)$ that computes $Q(\mathbf{z}) + O(\varepsilon)$.

Proof of Theorem 6.2. Let $\beta_1, \ldots, \beta_n$ be n distinct elements in $\mathbb{F}$ and let $\alpha_i := \mathbf{Esym}_i(\boldsymbol{\beta})$. Define

$$H(\mathbf{z}, y) := y^n - (z_1 + \alpha_1)y^{n-1} + (z_2 + \alpha_2)y^{n-2} - \cdots + (-1)^{n-1}(z_{n-1} + \alpha_{n-1})y + (-1)^n(z_n + \alpha_n).$$

Note that

$$\begin{aligned} H(\vec{0}, y) &= y^n - \alpha_1 y^{n-1} + \cdots + (-1)^n \alpha_n \\ &= (y - \beta_1) \cdots (y - \beta_n). \end{aligned}$$

Since $H(\vec{0}, y)$ is square-free, by Newton iteration (Theorem 4.12), there exist power series $A_i(\mathbf{z}) \in \mathbb{F}[\![\mathbf{z}]\!]$ such that $A_i(\vec{0}) = \beta_i$ and

$$H(\mathbf{z}, y) = \prod_{i=1}^{n} (y - A_i(\mathbf{z})).$$

Applying Theorem 5.6 with degree parameter D, we get a circuit C_1 that takes as input $z_1, \ldots, z_n, \beta_1, \ldots, \beta_n$ and outputs polynomials $\widetilde{A}_1, \ldots, \widetilde{A}_n \in \mathbb{F}[\mathbf{z}]$ that are degree-D approximations of the y-roots of $H(\mathbf{x}, y)$. Moreover, the circuit C_1 has size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$. Since $\{A_i(\mathbf{z})\}_{i=1}^n$ are the y-roots of $H(\mathbf{z}, y)$, by expanding the factorization of $H(\mathbf{z}, y)$, we see that $\mathbf{Esym}_j(A_1, \ldots, A_n) =$

$z_j + \alpha_j$ for all $j \in [n]$. Thus,

$$\begin{aligned} Q(z_1 + \alpha_1, \ldots, z_n + \alpha_n) &= P(A_1(\mathbf{z}), \ldots, A_n(\mathbf{z})) \\ &= P(A_1(\mathbf{z}), \ldots, A_n(\mathbf{z})) \bmod \langle \mathbf{z} \rangle^{D+1} \qquad \text{Since } \deg Q = D \\ &= P(\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z})) \bmod \langle \mathbf{z} \rangle^{D+1}. \end{aligned} \tag{6.3}$$

We build a circuit C_2 that feeds the approximate roots $\{\widetilde{A}_i(\mathbf{z})\}_{i=1}^n$ as inputs to C and outputs $P(\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z}))$. From Eq. (6.3), the output of C_2 is of the form

$$Q(z_1 + \alpha_1, \ldots, z_n + \alpha_n) + R(\mathbf{z}),$$

where every monomial of $R(\mathbf{z})$ has degree at least $D + 1$. Now, in the inputs to C_2, we replace z_i by εz_i and β_i by $\varepsilon\beta_i$. Since $\alpha_i = \mathbf{Esym}_i(\boldsymbol{\beta})$ and $\mathbf{Esym}_i$ is a homogeneous polynomial of degree i, this change of inputs sends α_i to $\varepsilon^i \alpha_i$. The output is now of the form

$$Q(\varepsilon z_1 + \varepsilon\alpha_1, \varepsilon z_2 + \varepsilon^2\alpha_2, \ldots, \varepsilon z_n + \varepsilon^n \alpha_n,) + R_i(\varepsilon z_1, \ldots, \varepsilon z_n).$$

Since Q is a homogeneous polynomial of degree D and every monomial of $R(\mathbf{z})$ has degree at least $D + 1$, the above expression simplifies to

$$\varepsilon^D Q(z_1 + \alpha_1, z_2 + \varepsilon\alpha_2, \ldots, z_n + \varepsilon^{n-1}\alpha_n) + O(\varepsilon^{D+1}).$$

We will divide the output by ε^D and call the resulting circuit C_3. The output of C_3 is of the form

$$Q(z_1 + \alpha_1, z_2 + \varepsilon\alpha_2, \ldots, z_n + \varepsilon^{n-1}\alpha_n) + O(\varepsilon).$$

Finally, we obtain the desired circuit C' by shifting z_i to $z_i - \varepsilon^{i-1}\alpha_i$. The resulting circuit C' outputs the polynomial $Q(\mathbf{z}) + O(\varepsilon)$. Note that from Corollary 4.6, there is an $\widetilde{O}(n)$ size and polylog(n) depth circuit for computing all $\{\alpha_i\}_{i=1}^n$ from $\{\beta_i\}_{i=1}^n$. Thus, the size of C' is the sum of the sizes of C and C_1 (the Newton iteration circuit), which is $\widetilde{O}(s + n \log D)$. The depth of C' is similarly the sum of the depths of C and C_1, which is $\Delta +$ polylog(n, D). Also, from Theorem 5.6, since there is an algorithm that outputs C_1 in $\widetilde{O}(n \log D)$ time, we have an algorithm to output the circuit C' in $\widetilde{O}(s + n \log D)$ time. □

For our applications, we need a slightly more general version of the preceding theorem. We need to handle

1. multi-output circuits instead of single-output circuits,
2. polynomials $P(\mathbf{x}, \mathbf{w})$ where P is symmetric only in the $\mathbf{x}$ variables, and
3. the case where the polynomial $Q(\mathbf{z}, \mathbf{w})$ is weighted homogeneous, not just homogeneous.

The proof of Theorem 6.2 almost immediately extends to this more general setting. The only change occurs when we multiply the $\mathbf{z}$ variables by ε to suppress the error term $R(\mathbf{z})$. Instead of multiplying each z_i by ε, we now multiply each z_i by different powers of ε, corresponding to the weights with respect to which the polynomial $Q(\mathbf{z}, \mathbf{w})$ is weighted homogeneous.

Theorem 6.4. *Let $\mathbb{F}$ be a field of size at least n and ε be a formal variable. Let $d_1, \ldots, d_m \in \mathbb{N}$ and $D = \max(d_1, \ldots, d_m)$. Let C be a multi-output algebraic circuit of size s and depth Δ, that outputs polynomials $P_1(\mathbf{x}, \mathbf{w}), \ldots, P_m(\mathbf{x}, \mathbf{w}) \in \mathbb{F}[\mathbf{x}, \mathbf{w}]$, which are symmetric in variables $x_1, \ldots, x_n$. For each $i \in [m]$, let $Q_i(z_1, \ldots, z_n, \mathbf{w}) \in \mathbb{F}[\mathbf{z}, \mathbf{w}]$ be the unique polynomial such that*

$$P_i(\mathbf{x}, \mathbf{w}) = Q_i(\mathbf{Esym}_1(\mathbf{x}), \ldots, \mathbf{Esym}_n(\mathbf{x}), \mathbf{w}).$$

Suppose for all $i \in [m]$, $Q_i(\mathbf{z}, \mathbf{w})$ is $(e_1, \ldots, e_n, \mathbf{f})$-homogeneous of weighted degree d_i. Then, there is a multi-output algebraic circuit C', defined over the field $\mathbb{F}(\varepsilon)$, of size $\widetilde{O}(s + n \log D)$ and depth $\Delta + \mathrm{polylog}(n, D)$, that outputs polynomials $Q_1(\mathbf{z}, \mathbf{w}) + O(\varepsilon), \ldots, Q_m(\mathbf{z}, \mathbf{w}) + O(\varepsilon)$. Moreover, there is an algorithm that, given D and the circuit C as input, outputs C' in $\widetilde{O}(s + n \log D)$ time.

Proof. Let $\beta_1, \ldots, \beta_n$ be n distinct elements in $\mathbb{F}$ and let $\alpha_i = \mathbf{Esym}_i(\boldsymbol{\beta})$. The proof largely follows the template of Theorem 6.2. Exactly as in Theorem 6.2, we define the polynomial $H(\mathbf{z}, y)$, its power series roots $A_1(\mathbf{z}), \ldots, A_n(\mathbf{z})$, and their degree-$D$ approximations $\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z})$, and build the circuit C_1 that computes the $\widetilde{A}_i(\mathbf{z})$ via Newton iteration. From Theorem 5.6 applied with degree parameter D, the circuit C_1 takes input $z_1, \ldots, z_n, \beta_1 \ldots, \beta_n$, outputs the approximate roots $\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z})$, and does so in size $\widetilde{O}(n \log D)$ and depth $\mathrm{polylog}(n, D)$. Since $\{A_i(\mathbf{z})\}_{i=1}^n$ are the y-roots of $H(\mathbf{z}, y)$, we have $\mathbf{Esym}_j(A_1, \ldots, A_n) = z_j + \alpha_j$ for all $j \in [n]$. Thus, for each $i \in [m]$, we have

$$\begin{aligned} Q_i(z_1 + \alpha_1, \ldots, z_n + \alpha_n, \mathbf{w}) &= P_i(A_1(\mathbf{z}), \ldots, A_n(\mathbf{z}), \mathbf{w}) \\ &= P_i(A_1(\mathbf{z}), \ldots, A_n(\mathbf{z}), \mathbf{w}) \bmod \langle \mathbf{z} \rangle^{D+1} \qquad \text{Since } \deg Q_i \le D \\ &= P_i(\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z}), \mathbf{w}) \bmod \langle \mathbf{z} \rangle^{D+1}. \end{aligned} \tag{6.5}$$

We build circuit C_2 that feeds in $\{\widetilde{A}_i(\mathbf{z})\}_{i=1}^n$ as inputs to C and outputs $\{P_i(\widetilde{A}_1(\mathbf{z}), \ldots, \widetilde{A}_n(\mathbf{z}), \mathbf{w})\}_{i=1}^m$. From Eq. (6.5), the i-th output of C_2 is of the form

$$Q_i(z_1 + \alpha_1, \ldots, z_n + \alpha_n, \mathbf{w}) + R_i(\mathbf{z}, \mathbf{w}), \tag{6.6}$$

where every monomial in $\mathbf{z}$ of $R(\mathbf{z}, \mathbf{w}) \in \mathbb{F}[\mathbf{w}][\mathbf{z}]$ has degree at least $D + 1$. Let $E := \max(e_1, \ldots, e_n)$. In the inputs to C_2, we replace z_j by $\varepsilon^{e_j} z_j$, β_j by $\varepsilon^E \beta_j$, and w_j by $\varepsilon^{f_j} w_j$. Since $\alpha_j = \mathbf{Esym}_j(\boldsymbol{\beta})$ and $\mathbf{Esym}_j$ is a degree-j homogeneous polynomial, α_j shifts to $\varepsilon^{jE} \alpha_j$. The output is now of the form

$$Q_i(\varepsilon^{e_1} z_1 + \varepsilon^E \alpha_1, \varepsilon^{e_2} z_2 + \varepsilon^{2E} \alpha_2, \ldots, \varepsilon^{e_n} z_n + \varepsilon^{nE} \alpha_n, \varepsilon^{\mathbf{f}} \mathbf{w}) + R_i(\varepsilon^{\mathbf{e}} \mathbf{z}, \varepsilon^{\mathbf{f}} \mathbf{w}).$$

Since Q_i is a weighted homogeneous polynomial of weighted degree d_i for weights $(e_1, \ldots, e_n, \mathbf{f})$ and every $\mathbf{z}$-monomial in $R_i(\mathbf{z}, \mathbf{w})$ has degree at least $D+1$, the above expression simplifies to

$$\varepsilon^{d_i} Q_i(z_1 + \varepsilon^{E-e_1}\alpha_1, z_2 + \varepsilon^{2E-e_2}\alpha_2, \ldots, z_n + \varepsilon^{nE-e_n}\alpha_n, \mathbf{w}) + O(\varepsilon^{D+1}). \tag{6.7}$$

We divide the i-th output by ε^{d_i}. We call the resulting circuit C_3. Its i-th output is of the form

$$Q_i(z_1 + \varepsilon^{E-e_1}\alpha_1, z_2 + \varepsilon^{2E-e_2}\alpha_2, \ldots, z_n + \varepsilon^{nE-e_n}\alpha_n, \mathbf{w}) + O(\varepsilon^{D-d_i+1})$$

Finally, we obtain the circuit C' by shifting z_i to $z_i - \varepsilon^{iE-e_i}\alpha_i$. The resulting circuit C' outputs polynomials $Q_1(\mathbf{z}, \mathbf{w}) + O(\varepsilon), \ldots, Q_m(\mathbf{z}, \mathbf{w}) + O(\varepsilon)$. Note that from Corollary 4.6, there is an $\widetilde{O}(n)$ size and polylog(n) depth circuit for computing all $\{\alpha_i\}_{i=1}^n$ from $\{\beta_i\}_{i=1}^n$. Thus, the size of C' is the sum of the sizes of C and C_1 (the Newton iteration circuit), which is $\widetilde{O}(s + n \log D)$. The depth of C' too is the sum of depths of C and C_1, which is $\Delta + \text{polylog}(n, D)$. Also, from Theorem 5.6, since there is an algorithm that outputs C_1 in $\widetilde{O}(n \log D)$ time, we have an algorithm to output circuit C' in $\widetilde{O}(s + n \log D)$ time. □

In all of our applications, we will be interested in designing small, shallow circuits that take the coefficients of a univariate polynomial $f = x^n + \sum_{i=0}^n f_i x^i$ as input. It turns out that these circuits are easy to design if we are given the roots $\alpha_1, \ldots, \alpha_n \in \overline{\mathbb{F}}$ of f as input instead of the coefficients. We will use Theorem 6.4 to transfer our circuit constructions from the root representation to the coefficient representation. An annoying technical detail is that the coefficient f_i of f is $(-1)^{n-i}\mathbf{Esym}_{n-i}(\boldsymbol{\alpha})$, not $\mathbf{Esym}_{n-i}(\boldsymbol{\alpha})$, and so we cannot apply Theorem 6.4 literally as written. A simple change of signs from z_j to $(-1)^i z_j$ in the polynomial $Q_i(\mathbf{z}, \mathbf{w})$ extends Theorem 6.4 to this setting, resulting in the following corollary.

Corollary 6.8. *Let $\mathbb{F}$ be a field of size at least n and ε be a formal variable. Let $d_1, \ldots, d_m \in \mathbb{N}$ and $D = \max(d_1, \ldots, d_m)$. Let C be a multi-output algebraic circuit of size s and depth Δ, that outputs polynomials $P_1(\mathbf{x}, \mathbf{w}), \ldots, P_m(\mathbf{x}, \mathbf{w}) \in \mathbb{F}[\mathbf{x}, \mathbf{w}]$, which are symmetric in variables $x_1, \ldots, x_n$. For each $i \in [m]$, let $Q_i(z_1, \ldots, z_n, \mathbf{w}) \in \mathbb{F}[\mathbf{z}, \mathbf{w}]$ be the unique polynomial such that*

$$P_i(\mathbf{x}, \mathbf{w}) = Q_i\left(\text{coeff}_y\left(\prod_{i=1}^n (y - x_i)\right), \mathbf{w}\right).$$

Suppose for all $i \in [m]$, $Q_i(\mathbf{z}, \mathbf{w})$ is $(e_1, \ldots, e_n, \mathbf{f})$-homogeneous of weighted degree d_i. Then, there is a multi-output algebraic circuit C', defined over the field $\mathbb{F}(\varepsilon)$, of size $\widetilde{O}(s + n \log D)$ and depth $\Delta + \text{polylog}(n, D)$, that outputs polynomials $Q_1(\mathbf{z}, \mathbf{w}) + O(\varepsilon), \ldots, Q_m(\mathbf{z}, \mathbf{w}) + O(\varepsilon)$. Moreover, there is an algorithm that, given D and the circuit C as input, outputs C' in $\widetilde{O}(s + n \log D)$ time.

Finally, we observe that Corollary 6.8 remains true if we assume that the symmetric polynomials $P(\mathbf{x})$ are themselves in the border of size-s circuits. This extension will be necessary for our GCD

algorithm in Section 9.

Corollary 6.9. *Let $\mathbb{F}$ be a field of size at least n and ε, δ be formal variables. Let $d_1, \ldots, d_m \in \mathbb{N}$ and $D = \max(d_1, \ldots, d_m)$. Let C be a multi-output algebraic circuit over $\mathbb{F}(\delta)$, of size s and depth Δ, that outputs polynomials $P_1(\mathbf{x}, \mathbf{w}) + O(\delta), \ldots, P_m(\mathbf{x}, \mathbf{w}) + O(\delta) \in \mathbb{F}(\delta)[\mathbf{x}, \mathbf{w}]$, where $P_1, \ldots, P_m$ are symmetric in variables $x_1, \ldots, x_n$. For each $i \in [m]$, let $Q_i(z_1, \ldots, z_n, \mathbf{w}) \in \mathbb{F}[\mathbf{z}, \mathbf{w}]$ be the unique polynomial such that*

$$P_i(\mathbf{x}, \mathbf{w}) = Q_i\left(\operatorname{coeff}_y\left(\prod_{i=1}^{n}(y - x_i)\right), \mathbf{w}\right).$$

Suppose for all $i \in [m]$, $Q_i(\mathbf{z}, \mathbf{w})$ is $(e_1, \ldots, e_n, \mathbf{f})$-homogeneous of weighted degree d_i. Then, there is a multi-output algebraic circuit C',defined over the field $\mathbb{F}(\varepsilon)$, of size $\widetilde{O}(s + n \log D)$ and depth $\Delta + \operatorname{polylog}(n, D)$, that outputs polynomials $Q_1(\mathbf{z}, \mathbf{w}) + O(\varepsilon), \ldots, Q_m(\mathbf{z}, \mathbf{w}) + O(\varepsilon)$. Moreover, there is an algorithm that, given D and the circuit C as input, outputs C' in $\widetilde{O}(s + n \log D)$ time.

Proof. The proof is almost identical to that of Theorem 6.4. The only change is that in Eq. (6.6), the output of the circuit is now

$$Q_i(z_1 + \alpha_1, \ldots, z_n + \alpha_n, \mathbf{w}) + R_i(\mathbf{z}, \mathbf{w}) + O(\delta).$$

Substituting δ by ε^{D+1} merges the $O(\delta)$ error term into the $O(\varepsilon^{D+1})$ error term appearing in Eq. (6.7). The rest of the proof remains the same as in Theorem 6.4. □

7 Resultant of $(f(x) + y)$ and $g(x)$

Let $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i \in \mathbb{F}[x]$ and $g(x) = x^m + \sum_{i=0}^{m-1} g_i x^i \in \mathbb{F}[x]$ be monic polynomials of degree n and m respectively. The Sylvester matrix of $f(x) + y$ and $g(x)$, denoted by $\operatorname{Syl}_x(f(x) + y, g(x))$, is the $(m + n) \times (m + n)$ matrix given by

$$\mathrm{Syl}_x(f(x)+y,g(x)) = \begin{pmatrix}
1 & & & & 1 & & & & \\
f_{n-1} & 1 & & & g_{m-1} & 1 & & & \\
\vdots & f_{n-1} & \ddots & & \vdots & g_{m-1} & \ddots & & \\
\vdots & \vdots & & 1 & g_1 & \vdots & \ddots & & \\
\vdots & \vdots & & f_{n-1} & g_0 & g_1 & & \ddots & \\
f_1 & \vdots & & \vdots & & g_0 & & & 1 \\
y+f_0 & f_1 & & \vdots & & & \ddots & & g_{m-1} \\
 & y+f_0 & & \vdots & & & & \ddots & \vdots \\
 & & \ddots & f_1 & & & & \ddots & g_1 \\
 & & & y+f_0 & & & & & g_0
\end{pmatrix}.$$

The resultant of $f(x)+y$ and $g(x)$, denoted by $\mathrm{Res}_x(f(x)+y,g(x))$, is equal to the determinant of $\mathrm{Syl}_x(f(x)+y,g(x))$. Clearly, $\mathrm{Res}_x(f(x)+y,g(x))$ is a polynomial in y with degree at most m. Given the coefficients of $f(x)$ and $g(x)$, we would like to compute the coefficients of $\mathrm{Res}_x(f(x)+y,g(x)) \in \mathbb{F}[y]$. Since the input size is $m+n$ and output size is m, it is feasible to ask for a near-linear-size circuit for computing the above resultant. Unfortunately, we are not able to get such a circuit, and instead show that $\mathrm{coeff}(\mathrm{Res}_x(f(x)+y,g(x)))$ is computable in the border of near-linear-size circuits.

We first observe that there is a simple circuit that computes $\mathrm{coeff}(\mathrm{Res}_x(f(x)+y,g(x)))$ given the roots of $g(x)$. We will use the Poisson formula for resultants below to build this circuit.

Lemma 7.1 (Poisson formula for resultants (see, e.g., [CLO05, Chapter 3]))**.** *Suppose $f(x), g(x) \in \mathbb{F}[x]$ are monic polynomials and let $\beta_1, \ldots, \beta_m \in \overline{\mathbb{F}}$ be the roots of $g(x)$. Then*

$$\mathrm{Res}_x(f,g) = \prod_{i=1}^{m} f(\beta_i).$$

Lemma 7.2 (Computing $\mathrm{Res}_x(f(x)+y,g(x))$ given roots of $g(x)$)**.** *Let $\mathbb{F}$ be a field. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}$, of size $\widetilde{O}(m+n)$ and depth $\mathrm{polylog}(m+n)$, such that for all monic polynomials $f(x)$ and $g(x)$ of degree equal to m and n respectively, we have*

$$C_{m,n}\left(\mathrm{coeff}(f), (\beta_1, \ldots, \beta_m)\right) = \mathrm{coeff}\left(\mathrm{Res}_x(f(x)+y,g(x))\right),$$

where $\beta_1, \ldots, \beta_m \in \overline{F}$ are the roots of $g(x)$.

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Proof. By the Poisson formula (Lemma 7.1), we have

$$\mathrm{Res}_x(f(x)+y, g(x)) = \prod_{i=1}^{m}(f(\beta_i)+y).$$

Using the above formula, we build the circuit C to compute $\mathrm{Res}_x(f(x)+y, g(x))$ as follows: using the circuit for univariate multipoint evaluation (Lemma 4.3), given $\mathrm{coeff}(f)$ and $\beta_1, \ldots, \beta_m$ as input, we compute $f(\beta_1), \ldots, f(\beta_m)$. Then using the circuit for multiplying linear terms (Corollary 4.5), given $f(\beta_1), \ldots, f(\beta_m)$ as input, we compute $\mathrm{coeff}(\prod_{i=1}^{n}(f(\beta_i)+y))$.

The size and depth bounds for C follow from Lemma 4.3 and Corollary 4.5. Also, again from Lemma 4.3 and Corollary 4.5, the above construction gives an algorithm, which given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$. □

Each coefficient of $\mathrm{Res}_x(f(x)+y, g(x))$ is a polynomial function of $\{f_i\}_{i=0}^{n-1}$ and $\{g_i\}_{i=0}^{m-1}$. Let $Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) := [y^i]\,\mathrm{Res}_x(f(x)+y, g(x))$ be the coefficient of y^i in $\mathrm{Res}_x(f(x)+y, g(x))$. We can write $\mathrm{Res}_x(f(x)+y, g(x))$ as

$$\mathrm{Res}_x(f(x)+y, g(x)) = y^m + \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) \cdot y^i.$$

Below, we observe that each $Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1})$ is a weighted homogeneous polynomial.

Lemma 7.3 ($\mathrm{Res}_x(f(x)+y, g(x))$ is weighted homogeneous)**.** *Let $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i$ and $g(x) = x^m + \sum_{i=0}^{m-1} g_i x^i$. Suppose*

$$\mathrm{Res}_x(f(x)+y, g(x)) = y^m + \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) \cdot y^i$$

where $Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) := [y^i]\,\mathrm{Res}_x(f(x)+y, g(x))$. Then, for each $i \in \{0, 1, \ldots, m-1\}$, Q_i is an $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous polynomial with weighted degree $n(m-i)$.

Proof. Let $\alpha_1, \ldots, \alpha_n$ be the roots of $f(x)$ and $\beta_1, \ldots, \beta_m$ be the roots of $g(x)$. By the Poisson formula (Lemma 7.1),

$$\begin{aligned}\mathrm{Res}_x(f(x)+y, g(x)) &= \prod_{i=1}^{m}(y+f(\beta_i)) \\ &= y^m + \sum_{i=0}^{m-1} \mathbf{Esym}_{m-i}(f(\beta_1), \ldots, f(\beta_m)) \cdot y^i.\end{aligned}$$

Let $P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m)$ be the coefficient of y^i above. Expanding out P_i, we have

$$\begin{aligned} P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) &= \mathbf{Esym}_{m-i}(f(\beta_1), \ldots, f(\beta_m)) \\ &= \mathbf{Esym}_{m-i}(\{f(\beta_j)\}_{j=1}^m) \\ &= \mathbf{Esym}_{m-i}(\{(\beta_j - \alpha_1) \cdot (\beta_j - \alpha_2) \cdots (\beta_j - \alpha_n)\}_{j=1}^m). \end{aligned}$$

Thus, $P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m)$ is a homogeneous polynomial of degree $n(m-i)$. By Lemma 4.10, it follows that Q_i is a $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous polynomial with weighted degree $n(m-i)$. □

From the circuit in Lemma 7.2 that takes the roots of $g(x)$ as input, we build an equivalent circuit (in the border sense) that takes coefficients of $g(x)$ as input. We use Theorem 6.4 to accomplish this. Note that Theorem 6.4 requires that $\mathrm{Res}_x(f(x) + y, g(x))$ is weighted homogeneous in the coefficients of $f(x)$ and $g(x)$, which we showed above in Lemma 7.3. We stitch together all of these details below.

Theorem 7.4 (Computing $\mathrm{Res}_x(f(x) + y, g(x))$ in border)**.** *Let $\mathbb{F}$ be a field and ε be a formal variable. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}(\varepsilon)$, of size $\widetilde{O}(m+n)$ and depth* polylog$(m+n)$*, such that for all monic polynomials $f(x)$ and $g(x)$ of degree equal to m and n respectively, we have*

$$C_{m,n}\left(\mathrm{coeff}(f), \mathrm{coeff}(g)\right) = \mathrm{coeff}\left(\mathrm{Res}_x(f(x) + y, g(x))\right) + O(\varepsilon).$$

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Proof. Suppose $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i$ and $g(x) = x^m + \sum_{i=0}^{m-1} g_i x^i \in \mathbb{F}[x]$ and $\beta_1, \ldots, \beta_m \in \overline{\mathbb{F}}$ are the roots of $g(x)$. By Lemma 7.2, there is a circuit C' of size $\widetilde{O}(m+n)$ and depth polylog(m, n) such that,

$$C'\left(\mathrm{coeff}(f), (\beta_1, \ldots, \beta_m)\right) = \mathrm{coeff}\left(\mathrm{Res}_x(f(x) + y, g(x))\right).$$

We will denote the outputs of C' by $\{P_i(f_0, \ldots, f_{n-1}, \beta_1, \ldots, \beta_m)\}_{i=0}^{m-1}$, so

$$\mathrm{Res}_x(f(x) + y, g(x)) = y^m + \sum_{i=0}^{m-1} P_i(f_0, \ldots, f_{n-1}, \beta_1, \ldots, \beta_m) \cdot y^i.$$

From Lemma 7.3, $\mathrm{Res}_x(f(x) + y, g(x))$ can also be written as

$$\mathrm{Res}_x(f(x) + y, g(x)) = y^m + \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) \cdot y^i,$$

where Q_i are weighted homogeneous polynomials of degree $n(m-i)$ for weights $(n-1, n-2, \ldots, 1, m-1, m-2, \ldots, 1)$. Since for all $i \in \{0, 1, \ldots, m-1\}$, the polynomial $P_i(f_0, \ldots, f_{n-1}, \beta_1, \ldots, \beta_m)$ is symmetric in $\boldsymbol{\beta}$, and Q_i is a weighted homogeneous polynomial of degree $n(m-i)$, we can apply Corollary 6.8 to the circuit C' to obtain a circuit C such that

$$C\left(\text{coeff}(f), \text{coeff}(g)\right) = \text{coeff}\left(\text{Res}_x(f(x)+y, g(x))\right) + O(\varepsilon).$$

The size and depth bounds for C follow from Lemma 7.2 and Corollary 6.8. Also, again from Lemma 7.2 and Corollary 6.8, the above construction gives an algorithm, which given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$. □

8 Modular Composition

Let $f(x)$, $g(x)$, and $h(x) \in \mathbb{F}[x]$. Given the coefficients of $f(x)$, $g(x)$, and $h(x)$, we would like to compute the coefficients of $f(g(x)) \bmod h(x)$. We assume that $f(x)$ and $g(x)$ have degree at most n and m, respectively, and that $h(x)$ has degree m. This assumption is without loss of generality, because $f(g(x)) = f(g(x) \bmod h(x)) \bmod h(x)$, and we can compute $g(x) \bmod h(x)$ (which has degree at most m) by using the circuit for polynomial division with remainder (Lemma 4.2). Since the input size is $O(m+n)$ and output size is m, we ask: Is there a near-linear-size circuit to compute $\text{coeff}(f(g(x)) \bmod h(x))$? Unfortunately, we are unable to answer this, and instead show that $\text{coeff}(f(g(x)) \bmod h(x))$ is computable in the border of near-linear-size circuits.

8.1 Modular Composition from roots of $h(x)$

Suppose $h(x)$ is square-free. As in Section 7, we first observe that there is a simple circuit to compute $\text{coeff}(f(g(x)) \bmod h(x))$ given the roots of $h(x)$. We will use the following lemma that shows that $\text{coeff}(f(g(x)) \bmod h(x))$ is uniquely determined by the evaluations of $f(g(x))$ at the roots of $h(x)$.

Lemma 8.1. *Let $f(x)$ and $g(x) \in \mathbb{F}[x]$. Let $h(x) \in \mathbb{F}[x]$ be a square-free polynomial with degree m and $\gamma_1, \ldots, \gamma_m \in \overline{\mathbb{F}}$ be the distinct roots of $h(x)$. Suppose $r(x)$ is the unique polynomial with degree less than m such that for all $i \in [m]$, $r(\gamma_i) = f(g(\gamma_i))$. Then $r(x) = f(g(x)) \bmod h(x)$.*

Proof. By the division algorithm,

$$f(g(x)) = q_0(x) \cdot h(x) + r_0(x),$$

where $q_0(x)$ is the quotient and $r_0(x)$ is the remainder with degree less than m. By definition of modular composition, $r_0(x) = f(g(x)) \bmod h(x)$. Note that for each $i \in [m]$, since $h(\gamma_i) = 0$, we

have $r_0(\gamma_i) = f(g(\gamma_i))$. Since $r(x)$ has degree less than m and by the uniqueness of the interpolating polynomial $r(x)$, we have $r(x) = r_0(x) = f(g(x)) \bmod h(x)$. □

Using the above lemma, we get the following simple, near-linear-size circuit for modular composition when we are given the roots of $h(x)$ as part of the input.

Lemma 8.2 (Circuit for Modular Composition given roots of $h(x)$)**.** *Let $\mathbb{F}$ be a field. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}$, such that the following is true. For all polynomials $f(x)$ and $g(x)$ of degree at most n and m, respectively, and all monic, square-free polynomials $h(x)$ of degree equal to m, we have,*

$$C_{m,n}\left(\text{coeff}(f), \text{coeff}(g), (\gamma_1, \ldots, \gamma_m)\right) = \text{coeff}\left(f(g(x)) \bmod h(x)\right),$$

where $\gamma_1, \ldots, \gamma_m \in \overline{\mathbb{F}}$ are the roots of $h(x)$. Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Proof. Let $r(x)$ be the unique interpolating polynomial with degree less than m such that for all $i \in [m]$, we have $r(\gamma_i) = f(g(\gamma_i))$. Lemma 8.1 implies $r(x) = f(g(x)) \bmod h(x)$. So, we build the circuit C to compute coeff(r) as follows: Using the circuit for univariate multipoint evaluation (Lemma 4.3), given coeff(g) and $\gamma_1, \ldots, \gamma_m$ as input, we compute $g(\gamma_1), \ldots, g(\gamma_m)$. Again using Lemma 4.3, given coeff(f) and $g(\gamma_1), \ldots, g(\gamma_m)$ as input, we compute $f(g(\gamma_1)), \ldots, f(g(\gamma_m))$. Finally, using the circuit for univariate interpolation (Lemma 4.4), given $(\gamma_1, f(g(\gamma_1))), \ldots, (\gamma_m, f(g(\gamma_m)))$ as input, we compute coeff(r).

The size and depth bounds for C follow from Lemma 4.3 and Lemma 4.4. Also, again from Lemma 4.3 and Lemma 4.4, the above construction gives an algorithm, which given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$. □

8.2 Homogeneity of Modular Composition

Suppose $h(x)$ is a monic polynomial. In this case, we observe below that the coefficients of $f(g(x)) \bmod h(x)$ are polynomial functions of coeff(f), coeff(g), and coeff(h).

Lemma 8.3. *Let $F(x)$ and $h(x) \in \mathbb{F}[x]$. Suppose $h(x)$ is a monic polynomial. Then the coefficients of the quotient and remainder of $F(x)$ divided by $h(x)$ are polynomial functions of* coeff(F) *and* coeff(h).

Proof. Suppose $\deg(F) = n$ and $\deg(h) = m$. Let $F(x) = \sum_{i=0}^{n} F_i x^i$ and $h(x) = x^m + \sum_{i=0}^{m-1} h_i x^i$. By the division algorithm, we have

$$F(x) = q(x) \cdot h(x) + r(x), \tag{8.4}$$

where $q(x)$ is the quotient and $r(x)$ is the remainder with degree less than m. By definition, we have $r(x) = F(x) \bmod h(x)$. If $\deg(F) < \deg(h)$, then $q(x) = 0$ and so $r(x) = F(x)$. Thus, the coefficients of $r(x)$ are trivially polynomial functions of $\mathrm{coeff}(F)$ and $\mathrm{coeff}(h)$.

Suppose instead that $\deg(F) \geq \deg(h)$, i.e., $n \geq m$. In this case, $\deg(q) = n - m$. Write $q(x) = \sum_{i=0}^{n-m} q_i x^i$ and $r(x) = \sum_{i=0}^{m-1} r_i x^i \in \mathbb{F}[x]$. By comparing the coefficients of x^i on both sides of Eq. (8.4), we obtain the following set of linear equations

$$\begin{pmatrix} 1 & & & & & & \\ h_{m-1} & \ddots & & & & & \\ \vdots & & 1 & & & & \\ \vdots & & h_{m-1} & 1 & & & \\ h_0 & & \vdots & & 1 & & \\ & \ddots & \vdots & & & \ddots & \\ & & h_0 & & & & 1 \end{pmatrix} \begin{pmatrix} q_{n-m} \\ \vdots \\ q_0 \\ r_{m-1} \\ \vdots \\ r_0 \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n-1} \\ F_{n-2} \\ \vdots \\ F_0 \end{pmatrix}.$$

Let us denote the above matrix by M. The matrix M is the Sylvester matrix of $h(x)$ and 1, where 1 is interpreted as a polynomial with degree $n - m + 1$. Since M is a lower triangular matrix with 1 along its diagonal entries, we have $\det(M) = 1$. Thus, M is invertible and we can solve for $\mathrm{coeff}(q) = (q_0, \ldots, q_{n-m})$ and $\mathrm{coeff}(r) = (r_0, \ldots, r_{m-1})$ from the above set of equations. Using Cramer's rule and the fact that $\det(M) = 1$, it follows that $(q_0, \ldots, q_{n-m})$ and $(r_0, \ldots, r_{m-1})$ are polynomial functions of $\mathrm{coeff}(F) = (F_0, \ldots, F_n)$ and $\mathrm{coeff}(h) = (h_0, \ldots, h_m)$. Thus, the coefficients of $q(x)$ and $r(x)$ are polynomial functions of $\mathrm{coeff}(F)$ and $\mathrm{coeff}(h)$. □

Suppose $f(x) = \sum_{i=0}^{n} f_i x^i$, $g(x) = \sum_{i=0}^{m} g_i x^i$ and $h(x) = x^m + \sum_{i=0}^{m-1} h_i x^i \in \mathbb{F}[x]$. Let

$$Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \coloneqq [x^i]\,(f(g(x)) \bmod h(x))$$

be the coefficient of x^i in $f(g(x)) \bmod h(x)$. We can write $f(g(x)) \bmod h(x)$ as

$$f(g(x)) \bmod h(x) = \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \cdot x^i.$$

Below, we will show that each $Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1})$ is a weighted homogeneous polynomial. Our proof makes use of the Lagrange interpolation formula, which we recall below.

Lemma 8.5 (Lagrange Interpolation (see, e.g., [vzGG13, Section 5.2])). *Let $\alpha_1, \ldots, \alpha_d$ be distinct points in $\mathbb{F}$. Then for any $\beta_1, \ldots, \beta_d \in \mathbb{F}$, there is a unique polynomial $r(x)$ of degree less than d such that*

$r(\alpha_i) = \beta_i$ *for all* $i \in [d]$. *Furthermore, the polynomial* $r(x)$ *is given by*

$$r(x) = \sum_{i=1}^{d} \beta_i \cdot \left(\prod_{j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j} \right).$$

To show that each Q_i is a weighted homogeneous polynomial, we use the same idea as in Lemma 4.9 and Lemma 4.10: if $h(x) \in \mathbb{F}[x]$ is a polynomial with roots $\gamma_1, \ldots, \gamma_m \in \overline{\mathbb{F}}$, then any homogeneous expression in the roots of h corresponds to a weighted homogeneous expression in the coefficients of h, since the coefficients of h are homogeneous functions (of known degree) of the roots of h. The same idea applies to expressions that are homogeneous in the roots of two polynomials, such as $g(\gamma_j)$, which can be expanded as

$$g(\gamma_j) = \prod_{i=1}^{m} (\gamma_j - \beta_m)$$

where $\beta_1, \ldots, \beta_m \in \overline{\mathbb{F}}$ are the roots of $g(x)$. If we let $\alpha_1, \ldots, \alpha_n \in \overline{\mathbb{F}}$ be the roots of $f(x)$, then the outer evaluation $f(g(\gamma_j))$ can be written as

$$f(g(\gamma_j)) = \prod_{i=1}^{n} (g(\gamma_j) - \alpha_i) = \prod_{i=1}^{n} \left(\prod_{k=1}^{m} (\gamma_j - \beta_k) - \alpha_i \right).$$

If we treat α_i as having degree m, then this is a homogeneous expression in all three sets of roots, and so the corresponding expression in the coefficients of f, g, and h is a weighted homogeneous polynomial.

We now make the preceding sketch precise, showing that each Q_i is a weighted homogeneous polynomial.

Lemma 8.6 (Modular composition is weighted homogeneous)**.** *Let* $f(x) = \sum_{i=0}^{n} f_i x^i$, $g(x) = \sum_{i=0}^{m} g_i x^i$ *and* $h(x) = x^m + \sum_{i=0}^{m-1} h_i x^i \in \mathbb{F}[x]$. *Suppose*

$$f(g(x)) \bmod h(x) = \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \cdot x^i,$$

where $Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \coloneqq [x^i](f(g(x)) \bmod h(x))$. *Then for each* $i \in \{0, 1, \ldots, m-1\}$, *the polynomial* Q_i *is an* $(m \cdot n, m \cdot (n-1), \ldots, m \cdot 0, m, m-1, \ldots, 0, m, m-1, \ldots, 1)$*-homogeneous polynomial with weighted degree* $mn - i$.

Proof. Let $\gamma_1, \ldots, \gamma_m \in \overline{\mathbb{F}}$ be the roots of $h(x)$, counted with multiplicity. Suppose $r(x)$ is the unique polynomial with degree less than m such that for all $i \in [m]$, we have $r(\gamma_i) = f(g(\gamma_i))$. By the

Lagrange Interpolation formula (Lemma 8.5), we have

$$r(x) = \sum_{j=1}^{m} f(g(\gamma_j)) \cdot \left(\prod_{k \neq j} \frac{x - \gamma_k}{\gamma_j - \gamma_k} \right). \tag{8.7}$$

Lemma 8.1 implies that $r(x) = f(g(x)) \bmod h(x)$, so for each $i \in \{0, 1, \ldots, m-1\}$, we have

$$\begin{aligned}
&Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \\
&= [x^i] r(x) \\
&= (-1)^{m-1-i} \sum_{j=1}^{m} f(g(\gamma_j)) \cdot \left(\prod_{k \neq j} \frac{1}{\gamma_j - \gamma_k} \right) \cdot \mathbf{Esym}_{m-1-i}(\gamma_1, \ldots, \gamma_{j-1}, \gamma_{j+1}, \ldots, \gamma_m).
\end{aligned} \tag{8.8}$$

The last equality above follows from Eq. (8.7). Let t be a fresh variable and consider the following auxiliary polynomials $F(x)$, $G(x)$, and $H(x)$ in $\mathbb{F}[t][x]$, defined as

$$\begin{aligned}
F(x) &:= \sum_{i=0}^{n} F_i x^i = \sum_{i=0}^{n} t^{m(n-i)} f_i x^i \\
G(x) &:= \sum_{i=0}^{m} G_i x^i = \sum_{i=0}^{m} t^{m-i} g_i x^i \\
H(x) &:= x^m + \sum_{i=0}^{m-1} H_i x^i = x^m + \sum_{i=0}^{m-1} t^{m-i} h_i x^i.
\end{aligned}$$

Note that for each $i \in [m]$, $H(t\gamma_i) = t^m h(\gamma_i) = 0$. Thus, $t\gamma_1, \ldots, t\gamma_m$ are the roots of $H(x)$. Also, note that $F(G(t\gamma_i)) = F(t^m g(\gamma_i)) = t^{mn} f(g(\gamma_i))$. Consider

$$\begin{aligned}
&Q_i(t^{mn} f_0, t^{m(n-1)} f_1, \ldots, t^0 f_n, t^m g_0, t^{m-1} g_1, \ldots, t^0 g_m, t^m h_0, t^{m-1} h_1, \ldots, t h_{m-1}) \\
&= Q_i(F_0, \ldots, F_n, G_0, \ldots, G_m, H_0, \ldots, H_{m-1}).
\end{aligned}$$

By applying Eq. (8.8) to $F(x)$, $G(x)$, and $H(x)$, we get,

$$\begin{aligned}
&Q_i(F_0, \ldots, F_{n-1}, G_0, \ldots, G_{m-1}, H_0, \ldots, H_{m-1}) \\
&= (-1)^{m-1-i} \sum_{j=1}^{m} F(G(t\gamma_j)) \cdot \left(\prod_{k \neq j} \frac{1}{t\gamma_j - t\gamma_k} \right) \cdot \mathbf{Esym}_{m-1-i}(t\gamma_1, \ldots, t\gamma_{j-1}, t\gamma_{j+1}, \ldots, t\gamma_m) \\
&= (-1)^{m-1-i} \sum_{j=1}^{m} t^{mn} f(g(\gamma_j)) \cdot \frac{1}{t^{m-1}} \left(\prod_{k \neq j} \frac{1}{\gamma_j - \gamma_k} \right) \cdot t^{m-1-i} \mathbf{Esym}_{m-i-1}(\gamma_1, \ldots, \gamma_{j-1}, \gamma_{j+1}, \ldots, \gamma_m) \\
&= t^{mn-i} (-1)^{m-1-i} \sum_{j=1}^{m} f(g(\gamma_j)) \cdot \left(\prod_{k \neq j} \frac{1}{\gamma_j - \gamma_k} \right) \cdot \mathbf{Esym}_{m-1-i}(\gamma_1, \ldots, \gamma_{j-1}, \gamma_{j+1}, \ldots, \gamma_m) \\
&= t^{mn-i} Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}, h_0, \ldots, h_{m-1})
\end{aligned}$$

From the above set of equations, and since Q_i is a polynomial (from Lemma 8.3), we have, for each $i \in \{0, 1, \ldots, m-1\}$, Q_i is $(m \cdot n, m \cdot (n-1), m \cdot (n-2), \ldots, m \cdot 0, m, m-1, \ldots, 0, m, m-1, \ldots, 1)$-homogeneous polynomial with weighted degree $mn - i$. □

8.3 Modular Composition in border from coefficients

In Lemma 8.2, we built a circuit for modular composition that takes the roots of $h(x)$ as input. We will use Theorem 6.4 to construct an equivalent circuit that takes coefficients of $h(x)$ as inputs and outputs the coefficients of $f(g(x)) \bmod h(x)$ in the border. Theorem 6.4 required that $f(g(x)) \bmod h(x)$ is weighted homogeneous in the coefficients of f, g, and h, which we have shown in Lemma 8.6.

Theorem 8.9 (Modular Composition in the border)**.** *Let $\mathbb{F}$ be a field and ε be a formal variable. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}(\varepsilon)$, such that the following is true. For all polynomials $f(x)$ and $g(x)$ of degree at most n and m, respectively, and all monic, square-free polynomials $h(x)$ of degree equal to m, we have,*

$$C_{m,n}\left(\text{coeff}(f), \text{coeff}(g), \text{coeff}(h)\right) = \text{coeff}\left(f(g(x)) \bmod h(x)\right) + O(\varepsilon).$$

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Proof. Suppose $f(x) = \sum_{i=0}^{n} f_i x^i$, $g(x) = \sum_{i=0}^{m} g_i x^i$, $h(x) = x^m + \sum_{i=0}^{m-1} h_i x^i \in \mathbb{F}[x]$ and $\gamma_1, \ldots, \gamma_m \in \overline{\mathbb{F}}$ are the roots of $h(x)$. By Lemma 8.2, there is a circuit C' of size $\widetilde{O}(m+n)$ and depth $\text{polylog}(m, n)$ such that

$$C'\left(\text{coeff}(f), \text{coeff}(g), (\gamma_1, \ldots, \gamma_m)\right) = \text{coeff}\left(f(g(x)) \bmod h(x)\right).$$

We will denote the outputs of C' by $\{P_i(f_0, \ldots, f_n, g_0, \ldots, g_m, \gamma_1, \ldots, \gamma_m)\}_{i=0}^{m-1}$, so

$$f(g(x)) \bmod h(x) = \sum_{i=0}^{m-1} P_i(f_0, \ldots, f_n, g_0, \ldots, g_m, \gamma_1, \ldots, \gamma_n) \cdot x^i$$

From Lemma 8.6, $f(g(x)) \bmod h(x)$ can also be written as,

$$f(g(x)) \bmod h(x) = \sum_{i=0}^{m-1} Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m, h_0, \ldots, h_{m-1}) \cdot x^i$$

where Q_i are weighted homogeneous polynomial of degree $nm - i$ for weights $(mn, m(n-1), m(n-2), \ldots, m, m, m-1, \ldots, 1,\ m, m-1, \ldots, 1)$. Since for all $i \in \{0, 1, \ldots, m-1\}$, the polynomial $P_i(f_0, \ldots, f_n, g_0, \ldots, g_m, \gamma_1, \ldots, \gamma_m)$ is symmetric in $\boldsymbol{\gamma}$, and Q_i is a weighted homogeneous polynomial of degree $nm - i$, we can apply Corollary 6.8 to the circuit C' to obtain a circuit C such

that

$$C\left(\text{coeff}(f), \text{coeff}(g), \text{coeff}(h)\right) = \text{coeff}\left(f(g(x)) \bmod h(x)\right) + O(\varepsilon).$$

The size and depth bounds for C follow from Lemma 8.2 and Corollary 6.8. Also, again from Lemma 8.2 and Corollary 6.8, the above construction gives an algorithm, which given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$. □

In Theorem 8.9, we had assumed that $h(x)$ is square-free. Below we will extend the theorem for non-square-free $h(x)$ by a simple continuity argument. For proving this, we will need the definition of the discriminant of a polynomial.

Definition 8.10. *Let $f(x) \in \mathbb{F}[x]$. We define the* discriminant of $f(x)$*, denoted by* $\text{Disc}(f)$*, to be equal to* $(-1)^{\binom{n}{2}} \text{Res}_x(f(x), f'(x))$. ◊

For a quadratic polynomial $p(x) = ax^2 + bx + c$, the discriminant has the familiar formula $\text{Disc}(p(x)) = a \cdot (b^2 - 4ac)$ and it is non-zero exactly when $p(x)$ is square-free. This property holds in general for all univariate polynomials.

Lemma 8.11 (see, e.g., [vzGG13, Corollary 6.17])**.** *A polynomial $f(x) \in \mathbb{F}[x]$ is square-free if and only if* $\text{Disc}(f) \neq 0$.

Having defined the discriminant, we generalize Theorem 8.9 for all monic polynomials $h(x)$.

Theorem 8.12. *Let $\mathbb{F}$ be a field and ε be a formal variable. For every $m, n \in \mathbb{N}$, there is a multi-output algebraic circuit $C_{m,n}$, defined over the field $\mathbb{F}(\varepsilon)$, such that the following is true. For all polynomials $f(x)$ and $g(x)$ of degree at most n and m respectively, and all monic polynomials $h(x)$ of degree equal to m, we have*

$$C_{m,n}\left(\text{coeff}(f), \text{coeff}(g), \text{coeff}(h)\right) = \text{coeff}\left(f(g(x)) \bmod h(x)\right) + O(\varepsilon).$$

Moreover, there is an algorithm that, given m and n as input, outputs $C_{m,n}$ in time $\widetilde{O}(m+n)$.

Proof. Suppose $h(x) = x^n + \sum_{i=0}^{m-1} h_i x^i \in \mathbb{F}[x]$. Let the coefficients of $f(g(x)) \bmod h(x)$ be of the form

$$f(g(x)) \bmod h(x) := \sum_{i=0}^{m-1} Q_i(h_0, \ldots, h_{m-1}) \cdot x^i \tag{8.13}$$

Note that Q_i also depends on $\text{coeff}(f)$ and $\text{coeff}(g)$, but for ease of notation, we do not specify it in the above expression.

Let ε be a formal variable. Consider the monic polynomial $\hat{h}(x) := r(\varepsilon, x) = (1-\varepsilon)h(x) + \varepsilon(x^n - 1) \in \mathbb{F}(\varepsilon)[x]$. Note that since $x^n - 1$ is square-free, by Lemma 8.11, we have $\text{Disc}(r(1, x)) \neq 0$. Thus

$\mathrm{Disc}(\hat{h}(x)) = \mathrm{Disc}(r(\varepsilon, x)) \neq 0$, and again by Lemma 8.11, this implies that $\hat{h}(x)$ is square-free. The coefficients of $\hat{h}(x)$ are of the form

$$\begin{aligned}\hat{h}(x) &= x^n + \sum_{i=1}^{m-1} (h_i - \varepsilon h_i) x^i + (h_0 - \varepsilon(h_0 - 1)) \\ &=: x^n + \sum_{i=0}^{m-1} \hat{h}_i x^i.\end{aligned}$$

By applying Eq. (8.13) to $f(x)$, $g(x)$, and $\hat{h}(x)$ (note that h and $\hat{h}$ have the same degree and are both monic), we have

$$f(g(x)) \bmod \hat{h}(x) := \sum_{i=0}^{m-1} Q_i(\hat{h}_0, \ldots, \hat{h}_{m-1}) \cdot x^i \tag{8.14}$$

Since $\hat{h}(x)$ is square-free and monic, by Theorem 8.9, there is a circuit C of size $\widetilde{O}(m+n)$ and depth $\mathrm{polylog}(m, n)$ such that

$$C\left(\mathrm{coeff}(f), \mathrm{coeff}(g), \mathrm{coeff}(\hat{h})\right) = \mathrm{coeff}\left(f(g(x)) \bmod \hat{h}(x)\right) + O(\varepsilon).$$

From Eq. (8.14), the outputs of C are $\{Q_i(\hat{h}_0, \ldots, \hat{h}_{m-1}) + O(\varepsilon)\}_{i=0}^{m-1}$. For each $i \in \{0, 1, \ldots, m-1\}$, we have

$$\begin{aligned}Q_i(\hat{h}_0, \ldots, \hat{h}_{m-1}) + O(\varepsilon) &= Q_i(h_0 + \varepsilon(h_0 - 1), h_1 + \varepsilon h_1, \ldots, h_{m-1} + \varepsilon h_{m-1}) + O(\varepsilon) \\ &= Q_i(h_0, h_1, \ldots, h_{m-1}) + O(\varepsilon) \\ &= [x^i](f(g(x)) \bmod h(x)) + O(\varepsilon) \qquad \text{(By Eq. (8.13))}\end{aligned}$$

The penultimate equality above follows by applying Taylor expansion around $(h_0, \ldots, h_{m-1})$. Thus, the circuit C outputs $\mathrm{coeff}(f(g(x)) \bmod h(x)) + O(\varepsilon)$. The size and depth bounds for C follow from Theorem 8.9. Also, Theorem 8.9 gives an algorithm that given m, n, outputs C in $\widetilde{O}(m+n)$ time. □

9 Fast & parallel polynomial GCD

Let $f(x)$ and $g(x)$ be monic polynomials of degree n and m, respectively, with $n \geq m$. We are interested in efficiently computing the GCD of $f(x)$ and $g(x)$.[6] In particular, we would like to construct a small, low-depth circuit that takes the coefficients of $f(x)$ and $g(x)$ as input and outputs the coefficients of $\gcd(f, g)$. In general, the GCD is not a rational function of the coefficients of $f(x)$

[6]The GCD is only defined up to scaling by a constant, so to speak of *the* GCD, we require the GCD to be a monic polynomial.

and $g(x)$, as the following example shows:

$$\gcd(x-\alpha, x-\beta) = \begin{cases} x-\alpha & \text{if } \alpha = \beta, \\ 1 & \text{if } \alpha \neq \beta. \end{cases}$$

Because of this, we cannot hope to compute the GCD from the coefficients of f and g with only the operations of addition, subtraction, multiplication, and division. Suppose that in addition to $\text{coeff}(f)$ and $\text{coeff}(g)$, we are given a natural number d with the promise that $d = \deg(\gcd(f,g))$. In this case, the coefficients of the GCD can be expressed as rational functions of the coefficients of f and g, as the following lemma shows.

Lemma 9.1. *Let $n, m, d \in \mathbb{N}$ be natural numbers such that $d \leq \min(n,m)$. There are rational functions $Q_0, \ldots, Q_d \in \mathbb{F}[y_0, \ldots, y_n, z_0, \ldots, z_m]$ such that for all polynomials $f, g \in \mathbb{F}[x]$ of degrees n and m, respectively, if $\deg(\gcd(f,g)) = d$, then*

$$Q_i(\text{coeff}(f), \text{coeff}(g)) = [x^i] \gcd(f(x), g(x))$$

for all $i \in \{0, 1, \ldots, d\}$.

Proof. Suppose $\deg(\gcd(f,g)) = d$. Then there exist polynomials $a(x)$ and $b(x)$ of degree less than $m-d$ and $n-d$, respectively, known as the *Bézout coefficients* of f and g, such that

$$a(x)f(x) + b(x)g(x) = \gcd(f,g).$$

By equating the coefficients of powers of x on the left- and right-hand sides above, we obtain a system of linear equations, where the coefficients are the known coefficients of $f(x)$ and $g(x)$, and the unknowns are the coefficients of $a(x)$ and $b(x)$. The polynomial on the left-hand side above has degree at most $n+m-d-1$. Because we are promised that $\deg(\gcd(f,g)) = d$, we can infer $n+m-2d$ equations: one that enforces the x^d term to have coefficient 1, and $n+m-2d-1$ equations that force all higher-degree terms in x to have a coefficient of zero. This results in the

linear system

$$\begin{pmatrix} f_n & & & & g_m & & & & & \\ f_{n-1} & f_n & & & g_{m-1} & g_m & & & & \\ \vdots & & \ddots & & \vdots & & \ddots & & & \\ f_{n-m+d+1} & \cdots & \cdots & f_n & g_{d+1} & \cdots & \cdots & g_m & & \\ \vdots & & & \vdots & \vdots & & & & \ddots & \\ f_{d+1} & \cdots & \cdots & f_m & g_{m-n+d+1} & \cdots & \cdots & \cdots & \cdots & g_m \\ \vdots & & & \vdots & \vdots & & & & & \vdots \\ \vdots & & & \vdots & \vdots & & & & & \vdots \\ f_{2d-m+1} & \cdots & \cdots & f_d & g_{2d-n+1} & \cdots & \cdots & \cdots & \cdots & g_d \end{pmatrix} \begin{pmatrix} a_{m-d-1} \\ \vdots \\ a_0 \\ b_{n-d-1} \\ \vdots \\ b_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

where the a_i and b_i are the coefficients of $a(x)$ and $b(x)$, respectively, and we adopt the convention that $f_i = g_i = 0$ if $i < 0$ above. If the matrix on the left-hand side above is invertible, then we can express the coefficients of $a(x)$ and $b(x)$ as rational functions in the coefficients of $f(x)$ and $g(x)$. Because $a(x)f(x) + b(x)g(x) = \gcd(f, g)$, this implies that the coefficients of the GCD are likewise rational functions of the coefficients of f and g.

It remains to show that the matrix above is invertible. Suppose it were not: then there would be polynomials $\hat{a}(x)$ and $\hat{b}(x)$ in its kernel, so the polynomial $\hat{a}(x)f(x) + \hat{b}(x)g(x)$ would have degree less than d. This contradicts the promise that $\deg(\gcd(f, g)) = d$, since $\gcd(f, g)$ is the lowest-degree polynomial in the ideal generated by f and g. Thus the matrix above is invertible as claimed.[7] □

From Lemma 9.1, we can hope to compute $\gcd(f, g)$ using an algebraic circuit if we also know its degree. Explicitly, we want to build a circuit family $\{C_{n,m,d}\}_{n,m,d \in \mathbb{N}}$ where $C_{n,m,d}$ takes as input $\text{coeff}(f)$ and $\text{coeff}(g)$, and if $\deg(\gcd(f, g)) = d$, then $C_{n,m,d}$ correctly outputs the coefficients of $\gcd(f, g)$. Borodin, von zur Gathen and Hopcroft [BvH82] described such a circuit family for computing the GCD. Their construction essentially follows the proof of Lemma 9.1. The main task is to invert the subresultant matrix appearing there, which can be done in polynomial size and $O(\log^2 n)$ depth due to work of Csanky [Csa76] and Berkowitz [Ber84].

Our goal is to show that the GCD can be computed in the border of circuits of near-linear size and polylogarithmic depth. Our main inspiration comes from recent work of [AW24] and [BKR+25], who showed that the GCD of two polynomials can be computed by constant-depth circuits of polynomial size. The key idea in these works was to manipulate the roots of polynomials $f(x)$ and $g(x)$ given implicit access to these roots via the coefficients of $f(x)$ and $g(x)$. Below, we

[7] The determinant of this matrix is known as a *subresultant* of f and g. More generally, this matrix is invertible whenever the extended Euclidean scheme of f and g contains a remainder of degree d. For details, see, e.g., [vzGG13, Corollary 6.49].

give a brief overview of [AW24] for computing the GCD of square-free polynomials, and then an extension of their technique to non-square-free polynomials in [BKR+25].

Suppose $\{\alpha_1, \ldots, \alpha_n\}$ is the multi-set of roots of f over $\overline{\mathbb{F}}$. Andrews & Wigderson [AW24] define the following operations to filter out the roots of $f(x)$ based on $g(x)$:

$$\text{Filter}(f \mid g \neq 0) := \prod_{i \in [n] \,:\, g(\alpha_i) \neq 0} (y - \alpha_i)$$
$$\text{Filter}(f \mid g = 0) := \prod_{i \in [n] \,:\, g(\alpha_i) = 0} (y - \alpha_i)$$

Suppose $f(x)$ and $g(x)$ are square-free polynomials. If we are given $\deg(\gcd(f, g))$, then [AW24] showed that the above filter operations can be computed by constant depth circuits. Note that the circuit takes as input $\text{coeff}(f)$ and $\text{coeff}(g)$, but does not have direct access to the roots $\{\alpha_1, \ldots, \alpha_n\}$. *A priori*, this seems surprising, since the filter operation definition itself seems to rely on knowledge of the roots $\{\alpha_1, \ldots, \alpha_n\}$. Andrews & Wigderson [AW24] showed that the filter operation can be computed essentially by evaluating the elementary symmetric polynomials on $\{g(\alpha_1), \ldots, g(\alpha_n)\}$. These evaluations are clearly symmetric in the α_i, and so by the fundamental theorem of symmetric polynomials, these evaluations can be computed from the coefficients of f and g. At this point, they observe that if $f(x)$ and $g(x)$ were square-free polynomials, then $\gcd(f, g) = \text{Filter}(f \mid g = 0)$.

Bhattacharjee et al. [BKR+25] generalized this idea to compute the GCD of arbitrary polynomials using the filtering operation. They consider the auxiliary polynomial $F(x, z) := f(x) + z \cdot g(x)$, where z is a fresh variable. Suppose $\{\alpha_1, \ldots, \alpha_n\}$ is now the multi-set of x-roots of $F(x, z)$ over the field $\overline{F(z)}$. They observed that $F(\alpha, z) = 0$ and $g(\alpha) = 0$ precisely when α is a root of $\gcd(f, g)$, and in such a case, the multiplicity of α as a root of $F(x, z)$ is the same as the multiplicity of α in $\gcd(f, g)$. In other words, they observed that

$$\gcd(f, g) = \text{Filter}(F \mid g = 0).$$

Our goal will be to show that $\text{Filter}(F \mid g = 0)$ is in the border of circuits of near-linear size and $\text{polylog}(m, n)$ depth. Before doing this, we first record as a lemma the fact that $\text{Filter}(F \mid g = 0)$ indeed equals $\gcd(f, g)$.

Lemma 9.2 (implicit in the proof of [BKR+25, Theorem 4.4])**.** *Let $f(x)$ and $g(x)$ be monic polynomials in $\mathbb{F}[x]$ of degree n and m, respectively, with $n \geq m$. Let $F(x, z) := f(x) + z \cdot g(x)$, where z is a fresh variable. Then*

$$\gcd(f(x), g(x)) = \text{Filter}(F \mid g = 0) = \frac{F(x, z)}{\text{Filter}(F \mid g \neq 0)},$$

where $F(x, z)$ is viewed as a polynomial in $\overline{\mathbb{F}(z)}[x]$.

From here on out, our main focus will be to compute $\text{Filter}(F \mid g \neq 0)$. From this expression, Lemma 9.2 above gives us a simple way to compute GCD using polynomial division with $F(x, z)$.

9.1 Computing filtered polynomials from roots

We start by implementing the filtering operation $\text{Filter}(f \mid g \neq 0)$. Although we only need to compute $\text{Filter}(F \mid g \neq 0)$ for $F(x, z) = f(x) + z \cdot g(x)$, we will address the general case of computing $\text{Filter}(f \mid g \neq 0)$ for any pair of polynomials f and g in this subsection. As in Sections 7 and 8, we will first design circuits that compute $\text{Filter}(f \mid g \neq 0)$ from the roots of $f(x)$ and $g(x)$. Unlike our previous results, we will only obtain a border circuit that computes $\text{Filter}(f \mid g \neq 0)$, as opposed to a circuit that exactly computes $\text{Filter}(f \mid g \neq 0)$.

We first determine how the coefficients of the polynomial $\text{Filter}(f \mid g \neq 0)$ depend on the roots of f and the coefficients of g. Because the polynomial $\text{Filter}(f \mid g \neq 0)$ is normalized to have leading coefficient one, its coefficients are rational functions of the roots of f and coefficients of g. These rational functions are easy to describe explicitly, and this description will be useful later when we want to show that they correspond to weighted homogeneous polynomials in the coefficients of f and g.

Lemma 9.3. *Let $\mathbb{F}$ be any field and let $n, m, d \in \mathbb{N}$ with $n \geq m \geq d$. Let $f, g \in \mathbb{F}[x]$ be monic polynomials of degrees n and m, respectively, and let $\{\alpha_1, \ldots, \alpha_n\} \subseteq \overline{\mathbb{F}}$ be the multi-set of roots of f. Suppose $\deg \text{Filter}(f \mid g \neq 0) = d$. Then the polynomial $\text{Filter}(f \mid g \neq 0)$ is given by*

$$\text{Filter}(f \mid g \neq 0) = \frac{\mathbf{Esym}_d((y - \alpha_1)g(\alpha_1), \ldots, (y - \alpha_n)g(\alpha_n))}{\mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n))}.$$

Proof. Let $h(x, y) := (y - x)g(x)$. For each $i \in [n]$, note that $h(\alpha_i, y) = 0$ if and only if $g(\alpha_i) = 0$. Since we know that $\deg \text{Filter}(f \mid g \neq 0) = d$, there are $n - d$ roots $\alpha_{i_1}, \ldots, \alpha_{i_{n-d}}$, counted with multiplicity, such that $h(\alpha_{i_j}, y) = 0$. This implies that

$$\begin{aligned}
\mathbf{Esym}_d(h(\alpha_1, y), \ldots, h(\alpha_n, y)) &= \prod_{i \in [n] : h(\alpha_i, y) \neq 0} h(\alpha_i, y) \\
&= \prod_{i \in [n] : g(\alpha_i) \neq 0} (y - \alpha_i) g(\alpha_i) \\
&= \left(\prod_{i \in [n] : g(\alpha_i) \neq 0} (y - \alpha_i) \right) \cdot \left(\prod_{i \in [n] : g(\alpha_i) \neq 0} g(\alpha_i) \right) \\
&= \text{Filter}(F \mid g \neq 0) \cdot \mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n)).
\end{aligned}$$

Thus

$$\text{Filter}(F \mid g \neq 0) = \frac{\mathbf{Esym}_d(h(\alpha_1, y), \ldots, h(\alpha_n, y))}{\mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n))},$$

as claimed. □

We now describe a small circuit that computes the numerator and denominator of $\mathrm{Filter}(f \mid g \neq 0)$ as determined in Lemma 9.3.

Lemma 9.4. *Let $\mathbb{F}$ be any field and let $n, m \in \mathbb{N}$ with $n \geq m$. There is a family of circuits $\{C_d : 0 \leq d \leq m\}$, defined over $\mathbb{F}(\varepsilon)$, such that for all $d \in \{0, 1, \ldots, m\}$, the circuit C_d has size $\widetilde{O}(n)$, depth $\mathrm{polylog}(n)$, and satisfies the following.*

Let $f, g \in \mathbb{F}[x]$ be monic polynomials of degrees n and m, respectively, and let $\{\alpha_1, \ldots, \alpha_n\} \subseteq \overline{\mathbb{F}}$ be the multi-set of roots of f. Suppose that $\deg(\mathrm{Filter}(f \mid g \neq 0)) = d$. Then the circuit C_d computes

$$\begin{aligned} &C_d((\alpha_1, \ldots, \alpha_n), \mathrm{coeff}(g)) \\ &\quad = \left(\mathbf{Esym}_d(\{g(\alpha_i)\}_{i=1}^n), \mathrm{coeff}\left(\mathbf{Esym}_d(\{(y - \alpha_i)g(\alpha_i)\}_{i=1}^n)\right)\right) + O(\varepsilon). \end{aligned}$$

Moreover, there is an algorithm that, given n, m, and d as input, outputs the circuit C_d in $\widetilde{O}(n)$ time.

Proof. We first compute $\mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n))$. From $\mathrm{coeff}(g)$ and $\alpha_1, \ldots, \alpha_n$, we compute the evaluations $g(\alpha_1), \ldots, g(\alpha_n)$ in $\widetilde{O}(n)$ size and $\mathrm{polylog}(n)$ depth using the multipoint evaluation circuit of Lemma 4.3. We can then compute $\mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n))$ from these evaluations within the same size and depth bounds using Corollary 4.6.

Let $h(x, y) \coloneqq (y - x)g(x)$. We now compute the coefficients of $\mathbf{Esym}_d(h(\alpha_1, y), \ldots, h(\alpha_n, y))$. This is where we make use of border complexity. Observe that

$$\begin{aligned} \prod_{i \in [n]} (\varepsilon + h(\alpha_i, y)) &= \sum_{j=0}^{n} \varepsilon^j \mathbf{Esym}_{n-j}(\{h(\alpha_i, y)\}_{i=1}^n) \\ &= \sum_{j=n-d}^{n} \varepsilon^j \mathbf{Esym}_{n-j}(\{h(\alpha_i, y)\}_{i=1}^n), \end{aligned}$$

where the second equality follows from the fact that $\deg(\mathrm{Filter}(f, g)) = d$, so the first $n - d - 1$ of the terms in the product $\prod_{i \in [n]} (\varepsilon + h(\alpha_i, y))$ simplify to ε. We can compute the y-coefficients of this polynomial by applying Corollary 4.5 to the product

$$\prod_{i=1}^{n} (\varepsilon + h(\alpha_i, y)) = \prod_{i=1}^{n} (\varepsilon + (y - \alpha_i) \cdot g(\alpha_i)).$$

We have already computed the $g(\alpha_i)$, so we can compute the y-coefficients of this polynomial in an additional $\widetilde{O}(n)$ size and $\mathrm{polylog}(n)$ depth. Finally, dividing the output of this computation by ε^{n-d} produces the y-coefficients of

$$\sum_{j=n-d}^{n} \varepsilon^{j-(n-d)} \mathbf{Esym}_{n-j}(\{h(\alpha_i, y)\}_{i=1}^n),$$

which tend to the y-coefficients of $\mathbf{Esym}_d(h(\alpha_1, y), \ldots, h(\alpha_n, y))$ as ε tends to zero.

Finally, we remark that the preceding circuit construction can be carried out in $\widetilde{O}(n)$ time, as each subcircuit invoked can be constructed in $\widetilde{O}(n)$ time, and the connecting gates between subcircuits can likewise be constructed in $\widetilde{O}(n)$ time. □

9.2 Weighted homogeneity of filtered polynomials

By combining Lemma 9.4 with Lemma 9.3, we obtain a small, low-depth circuit that computes the polynomial $\mathrm{Filter}(F \mid g \neq 0)$ when we have access to the roots of $F(x, z) \coloneqq f(x) + z \cdot g(x)$. This can be used to compute the GCD via Lemma 9.2. As in earlier sections, we will convert this to a circuit that computes $\mathrm{Filter}(F \mid g \neq 0)$ from the coefficients of F and g alone. To do this, we need to first establish that the coefficients of the filter polynomial $\mathrm{Filter}(f \mid g \neq 0)$ are weighted homogeneous functions of the coefficients of f and g.

Lemma 9.5 (Weighted homogeneity of filtered polynomials)**.** *Let $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i$ and $g(x) = x^m + \sum_{i=0}^{m-1} g_i x^i$. Suppose that $\deg(\mathrm{Filter}(f \mid g \neq 0)) = d$. Then there are polynomials $Q_d, \ldots, Q_0, Q_{den}$ such that*

$$\mathrm{Filter}(f \mid g \neq 0)(y) = \sum_{i=0}^{d} \frac{Q_i(f_0, \ldots, f_n, g_0, \ldots, g_m)}{Q_{den}(f_0, \ldots, f_n, g_0, \ldots, g_m)} y^i.$$

Moreover, for each $i \in [d]$, the polynomial Q_i is $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous of weighted degree $(m+1)d - i$, and the polynomial Q_{den} is $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous of weighted degree md.

Proof. Let $\{\beta_1, \ldots, \beta_m\} \subseteq \overline{\mathbb{F}}$ be the multi-set of roots of g. Consider the polynomials

$$P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) \coloneqq [y^i]\mathbf{Esym}_d((y - \alpha_1)g(\alpha_1), \ldots, (y - \alpha_n)g(\alpha_n))$$

and

$$P_{\mathrm{den}}(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) \coloneqq \mathbf{Esym}_d(g(\alpha_1), \ldots, g(\alpha_n)).$$

Recalling that $g(\alpha) = \prod_{i=1}^{m}(\alpha - \beta_i)$ is symmetric in the β_i, we see that the polynomials $P_d, \ldots, P_0, P_{\mathrm{den}}$ are symmetric in the α_i and β_i. By the fundamental theorem of symmetric polynomials (Theorem 6.1), there are polynomials $Q_d, \ldots, Q_0, Q_{\mathrm{den}}$ such that

$$\begin{aligned} Q_i(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) &= P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) \\ Q_{\mathrm{den}}(f_0, \ldots, f_{n-1}, g_0, \ldots, g_{m-1}) &= P_{\mathrm{den}}(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m). \end{aligned}$$

It is easy to see that P_{den} is a homogeneous polynomial of degree md, so by Lemma 4.10, we

conclude that Q_{den} is $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous of weighted degree md. To see that P_i is homogeneous, we expand it as

$$\begin{aligned} P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) &= [y^i]\mathbf{Esym}_d((y-\alpha_1)g(\alpha_1), \ldots, (y-\alpha_n)g(\alpha_n)) \\ &= [y^i]\left(\sum_{\substack{S \subseteq [n] \\ |S|=d}} \prod_{j \in S} (y-\alpha_j)g(\alpha_j)\right) \\ &= (-1)^{d-i} \sum_{\substack{S \subseteq [n] \\ |S|=d}} \sum_{\substack{T \subseteq S \\ |T|=i}} \prod_{j \in T} g(\alpha_j) \prod_{j \in S \setminus T} \alpha_j g(\alpha_j). \end{aligned}$$

Because $g(\alpha_j) = \prod_{k=1}^{m}(\alpha_j - \beta_m)$ has degree m, each term in the summation above has degree

$$m \cdot |T| + (m+1) \cdot |S \setminus T| = (m+1)d - i,$$

so $P_i(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m)$ is a homogeneous polynomial of degree $(m+1)d - i$. Lemma 4.10 implies that Q_i is $(n, n-1, \ldots, 1, m, m-1, \ldots, 1)$-homogeneous with weighted degree $(m+1)d - i$ as claimed. □

9.3 Computing the GCD in the border from coefficients

So far, we have seen small, shallow circuits that can compute $\text{Filter}(f \mid g \neq 0)$ in the border when the roots of f are given as an additional input. In this subsection, we will use Corollary 6.9 to build an equivalent circuit that computes $\text{Filter}(f \mid g \neq 0)$ when the coefficients of f and g are given as input. Once we have this, we can compute the GCD in small size and depth by an application of Lemma 9.2.

We start by computing $\text{Filter}(f \mid g \neq 0)$ when the coefficients of f and g are given as input.

Lemma 9.6. *Let $\mathbb{F}$ be any field and let $n, m \in \mathbb{N}$ with $n \geq m$. There is a family of circuits $\{C_d : 0 \leq d \leq m\}$, defined over $\mathbb{F}(\varepsilon)$, such that for all $d \in \{0, 1, \ldots, m\}$, the circuit C_d has size $\widetilde{O}(n)$, depth $\text{polylog}(n)$, and satisfies the following.*

Let $f, g \in \mathbb{F}[x]$ be monic polynomials of degrees n and m, respectively. Suppose that $\deg(\text{Filter}(f \mid g \neq 0)) = d$. Then the circuit C_d computes

$$C_d(\text{coeff}(f), \text{coeff}(g)) = \text{coeff}(\text{Filter}(f \mid g \neq 0)) + O(\varepsilon).$$

Moreover, there is an algorithm that, given n, m, and d as input, outputs the circuit C_d in $\widetilde{O}(n)$ time.

Proof. By Lemmas 9.3 and 9.4, there is a circuit C'_d of size $\widetilde{O}(n)$ and depth $\text{polylog}(n)$ that receives the multi-set $\{\alpha_1, \ldots, \alpha_n\} \subseteq \overline{\mathbb{F}}$ of roots of f as an additional input and border computes polynomials

$P_d, \dots, P_0, P_{\text{den}}$ such that for all i, we have

$$\frac{P_i(\alpha_1, \dots, \alpha_n, g_0, \dots, g_{m-1})}{P_{\text{den}}(\alpha_1, \dots, \alpha_n, g_0, \dots, g_{m-1})} = [y^i] \operatorname{Filter}(f \mid g \neq 0)(y).$$

Lemma 9.5 shows that there are weighted homogeneous polynomials $Q_d, \dots Q_0, Q_{\text{den}}$ such that

$$\begin{aligned} P_i(\alpha_1, \dots, \alpha_n, g_0, \dots, g_{m-1}) &= Q_i(f_0, \dots, f_{n-1}, g_0, \dots, g_{m-1}) \qquad \forall i \in \{0, 1, \dots, d\} \\ P_{\text{den}}(\alpha_1, \dots, \alpha_n, g_0, \dots, g_{m-1}) &= Q_{\text{den}}(f_0, \dots, f_{n-1}, g_0, \dots, g_{m-1}). \end{aligned}$$

Since the polynomials $P_d, \dots, P_0, P_{\text{den}}$ are symmetric in the α_i and the polynomials $Q_d, \dots, Q_0, Q_{\text{den}}$ are weighted homogeneous, all with respect to the same weights, we can apply Corollary 6.9 to C'_d. This yields a circuit C_d of size $\widetilde{O}(n)$ and depth $\operatorname{polylog}(n)$ that computes $Q_d, \dots, Q_0, Q_{\text{den}}$, and this circuit can be constructed in $\widetilde{O}(n)$ time. Dividing Q_i by Q_{den} produces the y^i-coefficient of $\operatorname{Filter}(f \mid g \neq 0)$, as desired. □

So far, we have built a near-linear-size and low-depth circuit that computes $\operatorname{Filter}(f \mid g \neq 0)$ in the border. Using Lemma 9.2, we can use one filtering operation and one polynomial division to compute the GCD of f and g.

Theorem 9.7. *Let $\mathbb{F}$ be any field and let $n, m \in \mathbb{N}$ with $n \geq m$. There is a family of circuits $\{C_d : 0 \leq d \leq m\}$, defined over $\mathbb{F}(\varepsilon)$, such that for all $d \in \{0, 1, \dots, m\}$, the circuit C_d has size $\widetilde{O}(n)$, depth $\operatorname{polylog}(n)$, and for all monic polynomials $f, g \in \mathbb{F}[x]$ of degrees n and m, respectively, such that $\deg(\gcd(f, g)) = d$, we have*

$$C_d(\operatorname{coeff}(f), \operatorname{coeff}(g)) = \operatorname{coeff}(\gcd(f, g)) + O(\varepsilon).$$

Moreover, there is an algorithm that, given n, m, and d as input, outputs the circuit C_d in $\widetilde{O}(n)$ time.

Proof. Let z be a fresh variable and define the polynomial $F(x, z) := f(x) + z \cdot g(x)$, which we view as an element of $\mathbb{F}(z)[x]$. From Lemma 9.2, we have

$$\gcd(f, g) = \operatorname{Filter}(F \mid g = 0) = \frac{F(x, z)}{\operatorname{Filter}(F \mid g \neq 0)}.$$

Because we assume $n \geq m$, it follows that $\deg(F) = n$. When $\deg(\gcd(f, g)) = d$, it must be the case that $\deg(\operatorname{Filter}(F \mid g \neq 0)) = n - d$. Applying Lemma 9.6, we obtain a circuit C' of size $\widetilde{O}(n)$ and depth $\operatorname{polylog}(n)$ that computes $\operatorname{coeff}(\operatorname{Filter}(F \mid g \neq 0)) + O(\varepsilon)$. We then use polynomial division with remainder (Lemma 4.2) to compute the coefficients of the quotient

$$h(x) := \frac{F(x, z)}{\operatorname{Filter}(F \mid g \neq 0) + O(\varepsilon)},$$

where the numerator and denominator are viewed as elements of $\mathbb{F}(z,\varepsilon)[x]$. It is clear that the resulting circuit has size $\widetilde{O}(n)$, depth polylog(n), and can be constructed in $\widetilde{O}(n)$ time. It remains to show that this correctly computes coeff$(\gcd(f,g)) + O(\varepsilon)$.

Recall Lemma 8.3 shows that when we divide two monic polynomials a and b, the coefficients of the quotient and remainder are polynomial functions of the coefficients of a and b. In particular, since Filter$(F \mid g \neq 0) + O(\varepsilon)$ is monic, the coefficients of $h(x)$ depend polynomially on the error term $O(\varepsilon)$. This implies that no divisions by ε occur in the coefficients of $h(x)$, so $h(x)|_{\varepsilon=0}$ is well-defined and equals $\frac{F(x,z)}{\text{Filter}(F|g\neq 0)} = \gcd(f,g)$. Thus $h(x) = \gcd(f,g) + O(\varepsilon)$ as desired. □

10 Open Problems

In this work, we showed that a variety of basic problems in computational algebra can be computed in the border of algebraic circuits of near-linear size and polylogarithmic depth. Of course, the main question left open by our work is to remove the use of border complexity from any of our algorithms and obtain small, shallow circuits that solve these problems exactly. This task may be particularly approachable for the polynomial GCD, a problem that can already be solved either in near-linear time or in constant depth. Even for modular composition and the bivariate resultant, it would be interesting to obtain near-linear-size circuits irrespective of their depth.

Key to all of our results was the ability to pass from circuits that receive the roots of a polynomial as input to circuits that receive a polynomial's coefficient as input while preserving the size and depth of the circuits. This relied on a fine-grained version of a result due to Bläser and Jindal [BJ19] on the complexity of symmetric polynomials. Is it possible to prove a variant of Theorem 6.2 that does not use border complexity? Such a result would be very interesting in its own right, and would immediately yield exact versions of the algorithms we design in this work.

In a complementary vein, what are the limits of small, shallow circuits? For example, we know that the resultant of two univariate polynomials can be computed by constant-depth algebraic circuits of polynomial size. Can these constant-depth circuits have near-linear size, or is it the case that any depth-Δ circuit for the resultant requires size $\Omega(n^{1+\varepsilon(\Delta)})$, where $\varepsilon(\Delta)$ is some function of Δ?

More generally, it would be interesting to better understand the role of border complexity in the design of algorithms for algebraic problems. To what extent should the results of this paper be viewed as evidence that modular composition can be solved in near-linear time? As mentioned in the introduction, border complexity upper bounds can be interpreted as a barrier towards proving lower bounds with known techniques. Should we interpret such barriers as evidence towards the existence of fast exact algorithms? Or is there a reasonable hypothesis under which some problem requires, say, quadratic time to solve exactly, but can be solved in near-linear time using border complexity?

Acknowledgements

We are thankful to Roshan Raj for insightful discussions at many stages of this work. We are also grateful to Swastik Kopparty and Ramprasad Saptharishi for many helpful discussions on the problems studied in this paper and for much encouragement.

References

[AW16] Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *computational complexity*, 25(1):217–253, Mar 2016.

[AW24] Robert Andrews and Avi Wigderson. Constant-Depth Arithmetic Circuits for Linear Algebra Problems. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2367–2386, 2024.

[BC88] Michael Ben-Or and Richard Cleve. Computing Algebraic Formulas Using a Constant Number of Registers. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 254–257, 1988.

[BCRL79] Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1997.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147 – 150, 1984.

[BFSS06] Alin Bostan, Philippe Flajolet, Bruno Salvy, and Éric Schost. Fast computation of special resultants. *Journal of Symbolic Computation*, 41(1):1–29, 2006.

[Bin80] Dario Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo*, 17:87–97, 1980.

[Bin84] Dario Bini. Parallel Solution of Certain Toeplitz Linear Systems. *SIAM Journal on Computing*, 13(2):268–276, 1984.

[BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On Algebraic Branching Programs of Small Width. *J. ACM*, 65(5), August 2018.

[BJ19] Markus Bläser and Gorav Jindal. On the Complexity of Symmetric Polynomials. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BK78] R. P. Brent and H. T. Kung. Fast Algorithms for Manipulating Formal Power Series. *Journal of the ACM*, 25(4):581–595, 1978.

[BKR+25] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. Constant-depth circuits for polynomial GCD over any characteristic, 2025. Pre-print available at `arXiv:2506.23220`.

[BM74] A. Borodin and R. Moenck. Fast modular transforms. *Journal of Computer and System Sciences*, 8(3):366–386, 1974.

[BP94] Dario Bini and Victor Y. Pan. *Polynomial and matrix computations, 1st Edition*, volume 12 of *Progress in theoretical computer science*. Birkhäuser, 1994.

[Bür04] Peter Bürgisser. The Complexity of Factors of Multivariate Polynomials. *Found. Comput. Math.*, 4(4):369–396, 2004.

[BvH82] Allan Borodin, Joachim von zur Gathen, and John Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982.

[CGGR23] Abhranil Chatterjee, Sumanta Ghosh, Rohit Gurjar, and Roshan Raj. Border Complexity of Symbolic Determinant Under Rank One Restriction. In *Proceedings of the 38th Annual Computational Complexity Conference (CCC 2023)*, volume 264 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CK91] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, Jul 1991.

[CLO05] David A. Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. Springer New York, NY, 2 edition, 2005.

[Csa76] L. Csanky. Fast parallel inversion algorithm. *SIAM Journal of Computing*, 5:618–623, 1976.

[DDS22] Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Demystifying the border of depth-3 algebraic circuits . In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, pages 92–103, Los Alamitos, CA, USA, February 2022. IEEE Computer Society.

[DGI+24] Pranjal Dutta, Fulvio Gesmundo, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Fixed-Parameter Debordering of Waring Rank. In *Proceedings of the 41st Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[DIK+24] Pranjal Dutta, Christian Ikenmeyer, Balagopal Komarath, Harshil Mittal, Saraswati Girish Nanoti, and Dhara Thakkar. On the Power of Border Width-2 ABPs over Fields of Characteristic 2. In *Proceedings of the 41st Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[DL25] Pranjal Dutta and Vladimir Lysikov. Recent Advances in Debordering Methods, 2025. Pre-print available at `arXiv:2510.13049`.

[DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the Roots: Uniform Closure Results for Algebraic Classes Under Factoring. *J. ACM*, 69(3), June 2022.

[FSV18] Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct Hitting Sets and Barriers to Proving Lower Bounds for Algebraic Circuits. *Theory of Computing*, 14(1):1–45, 2018.

[GJS21] Mark Giesbrecht, Armin Jamshidpey, and Éric Schost. Subquadratic-Time Algorithms for Normal Bases. *Comput. Complex.*, 30(1):5, 2021.

[GKSS17] Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. Pre-print available at `arXiv:1701.01717`.

[GMQ16] Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[IS22] Christian Ikenmeyer and Abhiroop Sanyal. A note on VNP-completeness and border complexity. *Information Processing Letters*, 176:106243, 2022.

[KL24] Yasunori Kinoshita and Baitian Li. Power Series Composition in Near-Linear Time . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2180–2185, Los Alamitos, CA, USA, October 2024. IEEE Computer Society.

[KS98] Erich Kaltofen and Victor Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comput.*, 67(223):1179–1197, 1998.

[KU08] Kiran S. Kedlaya and Christopher Umans. Fast Modular Composition in any Characteristic. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 146–155, 2008.

[Kum20] Mrinal Kumar. On the Power of Border of Depth-3 Arithmetic Circuits. *ACM Trans. Comput. Theory*, 12(1), February 2020.

[Kun74] H. T. Kung. On computing reciprocals of power series. *Numerische Mathematik*, 22(5):341–348, Oct 1974.

[NSSV24] Vincent Neiger, Bruno Salvy, Éric Schost, and Gilles Villard. Faster Modular Composition. *J. ACM*, 71(2), April 2024.

[Sho94] Victor Shoup. Fast Construction of Irreducible Polynomials over Finite Fields. *Journal of Symbolic Computation*, 17(5):371–391, 1994.

[Sho95] Victor Shoup. A New Polynomial Factorization Algorithm and its Implementation. *Journal of Symbolic Computation*, 20(4):363–397, 1995.

[Sho99] Victor Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, ISSAC '99, page 53–58, New York, NY, USA, 1999. Association for Computing Machinery.

[Shp25] Amir Shpilka. Improved Debordering of Waring Rank, 2025. Pre-print available at `arXiv:2502.03150`.

[Sie72] M. Sieveking. An algorithm for division of powerseries. *Computing*, 10(1):153–156, Mar 1972.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7(3):281–292, Sep 1971.

[Str73] Volker Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.*, 264:184–202, 1973.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il