Electronic Colloquium on Computational Complexity, Report No. 24 (2026)

Hilbert's Nullstellensatz is in the Counting Hierarchy

Robert Andrews* Abhibhav Garg† Éric Schost‡

Abstract

We show that Hilbert's Nullstellensatz, the problem of deciding if a system of multivariate polynomial equations has a solution in the algebraic closure of the underlying field, lies in the counting hierarchy. More generally, we show that the number of solutions to a system of equations can be computed in polynomial time with oracle access to the counting hierarchy. Our results hold in particular for polynomials with coefficients in either the rational numbers or a finite field. Previously, the best-known bounds on the complexities of these problems were PSPACE and FPSPACE, respectively. Our main technical contribution is the construction of a uniform family of constant-depth arithmetic circuits that compute the multivariate resultant.

Contents

*Cheriton School of Computer Science, University of Waterloo. 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.

†Cheriton School of Computer Science, University of Waterloo. Email: abhibhav.garg@uwaterloo.ca.

‡Cheriton School of Computer Science, University of Waterloo. Email: eschost@uwaterloo.ca.

ISSN 1433-8092

1 Introduction

1.1 Background

Polynomial equations are ubiquitous throughout mathematics and the sciences. They capture useful geometric and physical relationships, such as the Pythagorean theorem or the altitude of a projectile over time, and so have been the subject of investigation for centuries. The quadratic formula, Gaussian elimination, and the Euclidean algorithm, all important tools for solving equations, were known to ancient civilizations and are among the oldest algorithms to survive to the present day. Renaissance mathematicians later discovered formulas for the solution of cubic and quartic equations, and a great deal of effort was expended on the search for the quintic formula before Abel proved no such formula exists. The parallel development of numerical techniques, such as Newton's method, provided a means of solving high-degree equations even in the absence of an explicit formula for their solution, and they have since become an essential part of modern computational science.

Interest in solving polynomial equations has endured over time, and as computer science developed, mathematicians were naturally led to questions about the computability and complexity of solving equations. Already at the turn of the 20^{th} century, predating the computer age, Hilbert [Hil02] posed a list of twenty-three problems that might guide mathematical progress. Solving systems of equations was the subject of his tenth problem. In modern terms, Hilbert asked if there is an algorithm that takes as input polynomials $f_1, \ldots, f_m \in \mathbb{Z}[x_1, \ldots, x_n]$ and decides if the system of equations $f_1 = \cdots = f_m = 0$ has an integer solution. Matijasevič [Mat70], building on work of Davis, Putnam, and Robinson [DPR61], showed that this problem is undecidable. However, if we seek solutions in other domains, such as the real or complex numbers, this problem becomes decidable and has a different story to tell.

Over the complex numbers—and more generally, an algebraically closed field—the starting point for the solution of polynomial equations is Hilbert's Nullstellensatz. Given polynomials $f_1, \ldots, f_m \in \mathbb{C}[x_1, \ldots, x_n]$, the Nullstellensatz says that the system of equations $f_1 = \cdots = f_m = 0$ has no solution in $\mathbb{C}^n$ if and only if there are polynomials $g_1, \ldots, g_m \in \mathbb{C}[x_1, \ldots, x_n]$ such that

$$f_1(\mathbf{x})g_1(\mathbf{x}) + f_2(\mathbf{x})g_2(\mathbf{x}) + \cdots + f_m(\mathbf{x})g_m(\mathbf{x}) = 1.$$

Thus, we can decide if the system $f_1 = \cdots = f_m = 0$ has a solution by instead searching for polynomials $g_1, \ldots, g_m$ that satisfy $\sum_{i=1}^m f_i g_i = 1$. Because of the key role the Nullstellensatz plays in deciding the solvability of systems of equations, the computational problem of deciding whether a system of polynomial equations has a solution is likewise referred to as Hilbert's Nullstellensatz, which we abbreviate as HN.

The Nullstellensatz reduces our original nonlinear problem to a linear one: inspecting the coefficient of the monomial $\mathbf{x}^{\mathbf{a}}$ on both sides of the equation $\sum_{i=1}^m f_i g_i = 1$ yields a linear equation in the unknown coefficients of the polynomials $g_1, \ldots, g_m$. To turn this into an algorithm, we need to bound the degrees of the polynomials $g_1, \ldots, g_m$, should they exist. Hermann [Her26] was the first to do so. She showed that if the f_i have degree bounded by d and do not have a common solution, then one can find g_i of degree at most $(2d)^{n2^n}$.[1] This degree bound reduces the task of deciding the solvability of $f_1 = \cdots = f_m = 0$ to solving a system of linear equations of double-exponential size, a task that can be carried out in double-exponential time. Since then, numerous works have proved stronger bounds on the degrees (and heights) of the g_i, leading to bounds of the form $\deg(g_i) \leq d^n$ [Bro87; CGH88; Kol88; FG90; Som99; KPS01; Jel05]. Naturally, these single-exponential degree bounds imply that HN can be decided in exponential time, rather than double-exponential time, in the size of the input.

[1] The proofs in Hermann's original paper are incomplete, and the gaps were later filled by Seidenberg [Sei74].

Although degree bounds of the form $\deg(g_i) \leq d^n$ are essentially tight for the Nullstellensatz, there is more to say about its complexity, as linear equations can be solved in a surprisingly efficient manner. In 1976, Csanky [Csa76] designed an ingenious parallel algorithm to compute the determinant of an $n \times n$ matrix in $O(\log^2 n)$ time. The ensuing decade saw the design of parallel algorithms for many problems of linear algebra, including the algorithm of Borodin, von zur Gathen, and Hopcroft [BvH82] that solves linear systems in $O(\log^2 n)$ parallel time. Combined with an observation due to Borodin [Bor77] that parallel algorithms can be simulated in small space, this leads to an algorithm that solves linear systems using only $O(\log^2 n)$ space.[2] Scaled up to the exponentially-large systems of equations appearing in the Nullstellensatz (where the matrices have size $d^{\mathrm{poly}(n)}$), this results in a PSPACE algorithm to decide HN over an algebraically closed field, which is the state of the art for this problem.

Much less is known about the computational hardness of solving systems of equations. By arithmetizing SAT, it is easy to see that HN is NP-hard and that counting the number of solutions to a system of equations (if this number is guaranteed to be finite) is #P-hard. These are the strongest lower bounds known for the problem, and for good reason. Koiran [Koi96] showed that, assuming the Generalized Riemann Hypothesis, HN over the complex numbers is in the complexity class AM. Under plausible hardness assumptions, Miltersen and Vinodchandran [MV05] showed that AM = NP, so Koiran's work establishes that HN is conditionally NP-complete over $\mathbb{C}$. In light of this, any lower bound stronger than NP-hardness for HN would imply either a surprising collapse of complexity classes, or that one of the two conjectures underlying Koiran's NP algorithm is false.

Over other fields, the complexity of HN is less well understood. The aforementioned degree bounds are valid over any field, so the Nullstellensatz can be decided in PSPACE over any algebraically closed field, as long as the inputs lie in a subfield where arithmetic can be performed efficiently. Koiran's algorithm was recently extended by Ait El Manssour, Balaji, Nosan, Shirmohammadi, and Worrell [ABNSW25] to decide the Nullstellensatz over $\overline{\mathbb{C}(y_1, \ldots, y_k)}$ in AM, again assuming the Generalized Riemann Hypothesis. The situation is murkier over fields of positive characteristic. For example, over finite fields, we only know that HN is NP-hard and can be decided in PSPACE. No stronger lower bounds, nor better algorithms, are known, even conditionally.

Hilbert's Nullstellensatz, in addition to being a problem of intrinsic interest, also plays a central role in algebraic complexity, similar to the one occupied by boolean satisfiability in classical complexity theory. Over a commutative ring R, the problem HN_R—that is, deciding whether a system of polynomial equations over R has a solution in R—is complete for NP_R, the analogue of NP in the Blum–Shub–Smale model of computation [BSS89; BCSS98]. The counting variant of HN_R, denoted by $\#\mathsf{HN}_R$, where one must count the number of solutions to a system of equations over a ring R, is similarly important in the theory of counting problems. Bürgisser and Cucker [BC06] defined the complexity classes $\#\mathsf{P}_{\mathbb{R}}$ and $\#\mathsf{P}_{\mathbb{C}}$, the counting versions of $\mathsf{NP}_{\mathbb{R}}$ and $\mathsf{NP}_{\mathbb{C}}$, respectively, and showed that $\#\mathsf{HN}_{\mathbb{R}}$ and $\#\mathsf{HN}_{\mathbb{C}}$ are complete for their respective classes. The Nullstellensatz is also a useful algorithmic primitive in computational algebraic geometry: many problems of interest, such as computing the dimension of an algebraic variety over an algebraically closed field [Koi97], use the Nullstellensatz as an essential subroutine.

1.2 Our results

We show that a variety of problems related to solving systems of polynomial equations, chief among them the Nullstellensatz, can be decided in CH, the counting hierarchy. We work in the Turing

[2] The algorithms of Csanky [Csa76] and Borodin, von zur Gathen, and Hopcroft [BvH82] were stated as arithmetic algorithms, without regard to bit complexity, but they can likewise be implemented in $O(\log^2 n)$ parallel time when accounting for bit complexity.

machine model, representing polynomials by the binary encoding of their coefficients. Throughout, we consider systems of equations that have coefficients in the rational numbers, a number field, a finite field, or a polynomial ring over such a field, all of which can be efficiently encoded in binary, and are interested in their solvability over the algebraic closure of the coefficient field.

We use the dense representation of polynomials, where all monomials and their coefficients are listed, including monomials with a coefficient of zero. One could also consider more expressive encodings of a multivariate polynomial, either writing only the nonzero monomials of the polynomial (the *sparse representation*) or describing an arithmetic circuit that computes it (the *straight-line program representation*). By adding polynomially-many extension variables, both the sparse and straight-line program representations can be converted to the dense representation. This conversion preserves both the satisfiability and number of solutions (if this number is finite) of the original system of equations.

Our main result is that Hilbert's Nullstellensatz—that is, whether a system of equations over a field $\mathbb{F}$ has a solution in the algebraic closure $\overline{\mathbb{F}}$—can be decided in CH.

Theorem 1.1 (see Theorem 6.9)**.** *Let $\mathbb{F}$ be one of the fields $\mathbb{Q}$, $\mathbb{Q}(y_1, \ldots, y_k)$, a number field $\mathbb{K}$, $\mathbb{K}(y_1, \ldots, y_k)$, the finite field $\mathbb{F}_q$, or $\mathbb{F}_q(y_1, \ldots, y_k)$. Then Hilbert's Nullstellensatz over $\mathbb{F}$ can be decided in CH.*

Before proceeding to our other results, we pause briefly to recall the counting hierarchy. The counting hierarchy CH was first defined by Wagner [Wag86] to characterize the complexity of combinatorial counting problems. The k^{th} level of CH, denoted $\mathsf{C}_k\mathsf{P}$, is defined inductively as

$$\begin{aligned} \mathsf{C}_0\mathsf{P} &\coloneqq \mathsf{P} \\ \mathsf{C}_1\mathsf{P} &\coloneqq \mathsf{PP} \\ \mathsf{C}_k\mathsf{P} &\coloneqq \mathsf{PP}^{\mathsf{C}_{k-1}}. \end{aligned}$$

The counting hierarchy is the union of these classes, i.e., $\mathsf{CH} \coloneqq \bigcup_{k \geq 0} \mathsf{C}_k\mathsf{P}$. Alternatively, one can define CH as the class of languages that are decidable by polynomial-time Turing machines with a constant number of polynomially-bounded majority quantifiers, analogous to how PH captures languages that are decidable in polynomial time by Turing machines that may use a constant number of polynomially-bounded existential and universal quantifiers.

Where does the counting hierarchy sit in the landscape of complexity classes? Toda's result that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}$ [Tod91] shows that the polynomial-time hierarchy is contained in the second level of CH, so we have the inclusion $\mathsf{PH} \subseteq \mathsf{CH}$. It is also easy to see that $\mathsf{CH} \subseteq \mathsf{PSPACE}$ as a consequence of $\mathsf{PP} \subseteq \mathsf{PSPACE}$. As the name suggests, the counting hierarchy captures the complexity of counting: as a consequence of the definition, if $\mathsf{FP} = \#\mathsf{P}$ (i.e., if the permanent of a matrix can be computed in polynomial time), then we have the collapse $\mathsf{P} = \mathsf{CH}$. For more details on the counting hierarchy, we refer the reader to Allender and Wagner [AW90] and Torán [Tor91].

Returning to our results, Hilbert's Nullstellensatz shows that a system of polynomial equations $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ is unsatisfiable if and only if the constant polynomial 1 is in $\operatorname{rad}(f_1, \ldots, f_m)$, the radical of the ideal generated by $f_1, \ldots, f_m$. More generally, the Nullstellensatz says that a polynomial $h \in \mathbb{F}[x_1, \ldots, x_n]$ vanishes on the common zeroes of $f_1, \ldots, f_m$, called the *variety* of $f_1, \ldots, f_m$ and denoted by $\mathrm{V}(f_1, \ldots, f_m)$, if and only if $h \in \operatorname{rad}(f_1, \ldots, f_m)$. The problem of deciding if $h \in \operatorname{rad}(f_1, \ldots, f_m)$, known as the *radical ideal membership problem*, is another basic problem in computational algebraic geometry. The previously-mentioned degree bounds for the Nullstellensatz also apply to the radical ideal membership problem, with an additional factor to account for the degree of the polynomial h. These degree bounds imply that radical ideal membership, like HN, can

be solved in PSPACE. By combining our main theorem with the Rabinowitsch trick, we improve this bound, showing that radical ideal membership can be decided in CH.

Theorem 1.2 (see Corollary 7.1)**.** *Let $\mathbb{F}$ be one of the fields $\mathbb{Q}$, $\mathbb{Q}(y_1, \ldots, y_k)$, a number field $\mathbb{K}$, $\mathbb{K}(y_1, \ldots, y_k)$, the finite field $\mathbb{F}_q$, or $\mathbb{F}_q(y_1, \ldots, y_k)$. Then the radical ideal membership problem over $\mathbb{F}$ can be decided in* CH.

In contrast, Mayr and Meyer [MM82] proved that the general ideal membership problem is EXPSPACE-complete. Because of this, a similar improvement in the complexity of the ideal membership problem would apply to the class EXPSPACE itself.

Once we know that a system of equations $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ has a solution, a natural next step is to understand how many solutions this system has. One way to make this question precise is to ask for the dimension of the variety $\mathrm{V}(f_1, \ldots, f_m)$, which is a coarse measure of the size of the solution set. A straightforward application of our main theorem lets us compute this dimension in $\mathsf{FP}^{\mathsf{CH}}$.

Theorem 1.3 (see Corollary 7.2)**.** *Let $\mathbb{F}$ be one of the fields $\mathbb{Q}$, $\mathbb{Q}(y_1, \ldots, y_k)$, a number field $\mathbb{K}$, $\mathbb{K}(y_1, \ldots, y_k)$, the finite field $\mathbb{F}_q$, or $\mathbb{F}_q(y_1, \ldots, y_k)$. Given a set of polynomials $f_1, \ldots, f_m$ in $\mathbb{F}[x_1, \ldots, x_n]$, one can compute the dimension of the variety $\mathrm{V}(f_1, \ldots, f_m)$ in $\mathsf{FP}^{\mathsf{CH}}$.*

When the variety $\mathrm{V}(f_1, \ldots, f_m)$ is zero-dimensional, it consists of a finite set of points, so it makes sense to ask for the number of solutions to the system $f_1 = \cdots = f_m = 0$. Unlike the last two applications of our main theorem, which follow by simple reductions to HN, we are not aware of a simple reduction from counting points on a variety to HN. Despite this, our techniques easily extend to this problem, allowing us to compute the number of points on a variety in $\mathsf{FP}^{\mathsf{CH}}$.

Theorem 1.4 (see Theorem 6.16)**.** *Let $\mathbb{F}$ be one of the fields $\mathbb{Q}$, $\mathbb{Q}(y_1, \ldots, y_k)$, a number field $\mathbb{K}$, $\mathbb{K}(y_1, \ldots, y_k)$, the finite field $\mathbb{F}_q$, or $\mathbb{F}_q(y_1, \ldots, y_k)$. Given a set of polynomials $f_1, \ldots, f_m$ in $\mathbb{F}[x_1, \ldots, x_n]$, one can compute the cardinality of the variety $\mathrm{V}(f_1, \ldots, f_m)$ in $\mathsf{FP}^{\mathsf{CH}}$.*

Counting points on a variety is a basic problem of counting complexity whose precise classification is not well understood. Bürgisser and Cucker [BC06] introduced the complexity class GCC, for *geometric counting complex problems*, and showed that counting points on a complex variety is complete for GCC.[3] The class GCC and its functional analogue $\mathsf{FP}^{\mathsf{GCC}}$ contain natural counting problems in algebraic geometry, including computing the geometric degree of a variety [BC06], the Euler characteristic of a variety [BCL05], and the Hilbert polynomial of a smooth equidimensional variety [BL07]. By arithmetizing #SAT, it is easy to see that counting points on a variety is at least as hard as counting the number of satisfying assignments to a boolean formula, so $\#\mathsf{P} \subseteq \mathsf{GCC}$. Prior to our work, the best known upper bound on GCC was $\mathsf{GCC} \subseteq \mathsf{FPSPACE}$ [BC06]. Theorem 1.4 improves this to $\mathsf{GCC} \subseteq \mathsf{FP}^{\mathsf{CH}}$.

Theorem 1.4 explains why our techniques only prove an upper bound of CH on the complexity of HN, as opposed to placing HN in the polynomial-time hierarchy. The same techniques underlie the proofs of Theorems 1.1 and 1.4. Because Theorem 1.4 addresses a #P-hard problem, the best upper bound we can hope to prove with these techniques—excepting a surprising collapse of complexity classes—is #P.

Finally, we mention a straightforward application of Theorem 1.1 to computing tensor rank, an important problem in algebraic complexity related to the determination of the exponent of matrix multiplication. By a direct reduction to HN, we can compute the rank of a given tensor in $\mathsf{FP}^{\mathsf{CH}}$.

[3] Bürgisser and Cucker [BC06] also introduced the counting classes $\#\mathsf{P}_{\mathbb{C}}$ and $\#\mathsf{P}_{\mathbb{R}}$ as analogues of #P in the Blum–Shub–Smale model of computation. Among other results, they proved that counting points on complex or real varieties are complete problems for $\#\mathsf{P}_{\mathbb{C}}$ and $\#\mathsf{P}_{\mathbb{R}}$, respectively.

Theorem 1.5 (see Corollary 7.3)**.** *Let $\mathbb{F}$ be one of the fields $\mathbb{Q}$, $\mathbb{Q}(y_1, \ldots, y_k)$, a number field $\mathbb{K}$, $\mathbb{K}(y_1, \ldots, y_k)$, the finite field $\mathbb{F}_q$, or $\mathbb{F}_q(y_1, \ldots, y_k)$. Given a tensor $T \in \mathbb{F}^{d_1} \otimes \cdots \otimes \mathbb{F}^{d_k}$, one can compute the tensor rank of T over $\overline{\mathbb{F}}$ in $\mathsf{FP}^{\mathsf{CH}}$.*

1.3 Proof overview

As we saw, we can decide if a system of polynomials $f_1, \ldots, f_m \in \mathbb{C}[x_1, \ldots, x_n]$ of degrees at most d has a solution by deciding if the linear system

$$f_1(\mathbf{x})g_1(\mathbf{x}) + f_2(\mathbf{x})g_2(\mathbf{x}) + \cdots + f_m(\mathbf{x})g_m(\mathbf{x}) = 1.$$

is solvable, where the unknowns are the coefficients of the polynomials $g_1, \ldots, g_m \in \mathbb{C}[x_1, \ldots, x_n]$ of degrees at most d^n, which results in a system of equations of size $d^{\mathrm{poly}(n)}$. To improve the complexity of HN, we will take advantage of the fact that these linear systems are not arbitrary, but instead highly structured.

1.3.1 The resultant

To see what makes linear systems like $\sum_{i=1}^m f_i g_i = 1$ special, let us consider the problem of deciding if a system of two equations $f_1, f_2 \in \mathbb{C}[x]$ in one variable has a solution. The determinant of the linear system $f_1(x)g_1(x) + f_2(x)g_2(x) = 1$ is a well-known function of the coefficients of f_1 and f_2 called their *resultant*, denoted by $\mathrm{Res}(f_1, f_2)$. The resultant has the remarkable property that $\mathrm{Res}(f_1, f_2) = 0$ if and only the system $f_1(x) = f_2(x) = 0$ has a solution. By definition, the resultant is the determinant of a polynomially-large matrix, so it can be computed in polylogarithmic space using Csanky's algorithm [Csa76] for the determinant. This corresponds to a scaled-down version of the PSPACE algorithm for HN.

Although the resultant is defined as a determinant, the resultant can be computed more efficiently than the determinant itself. Andrews and Wigderson [AW24] showed that the resultant of two polynomials can be computed by arithmetic circuits of constant depth and polynomial size, something which is provably impossible for the determinant [LST25]. Because iterated addition and multiplication of rational numbers can be computed by threshold circuits of constant depth and polynomial size, i.e., in the class TC^0 [HAM02], this implies that the resultant can also be computed in TC^0 when the input is represented in binary. As we will soon see, the class TC^0 corresponds to a scaled-down version of the counting hierarchy CH, so this algorithm is evidence that it may be possible to improve the complexity of HN. The improved algorithm for the resultant relies on the *Poisson formula*, which expresses $\mathrm{Res}(f_1, f_2)$ in terms of the complex roots of f_1 and f_2. Suppose f_1 and f_2 are monic, and that f_1 factors as $f_1(x) = \prod_{i=1}^d (x - \alpha_i)$ over $\mathbb{C}$. In this case, the Poisson formula expresses the resultant of f_1 and f_2 as

$$\mathrm{Res}(f_1, f_2) = \prod_{i=1}^{d} f_2(\alpha_i).$$

Although not immediately obvious, this identity can be used to compute the resultant in constant depth, since the coefficients of f_1 provide useful information about its roots $\alpha_1, \ldots, \alpha_d$.

The resultant can be generalized to many polynomials in several variables, and designing algorithms to compute it will be essential for our work. Suppose $F_0, \ldots, F_N \in \mathbb{C}[x_0, \ldots, x_n]$ are homogeneous polynomials of degrees $d_0, \ldots, d_n$, respectively. There is a polynomial function of their coefficients, likewise called the resultant and denoted $\mathrm{Res}(F_0, \ldots, F_n)$, such that $\mathrm{Res}(F_0, \ldots, F_n) = 0$

exactly when the system $F_0(\mathbf{x}) = \cdots = F_n(\mathbf{x}) = 0$ has a nonzero solution.[4] Thus, if we can compute the resultant efficiently, we can also decide HN, at least in the case of homogeneous systems where the number of polynomials and variables match. The multivariate resultant can be expressed as the quotient of two minors of a structured matrix of size $\binom{d_0+\cdots+d_n}{n}$, called the *Macaulay matrix* of $F_0, \ldots, F_n$. At worst, this matrix is exponentially large compared to $\sum_{i=0}^{n} \binom{n+d_i}{n}$, the number of coefficients of the polynomials $F_0, \ldots, F_n$, so any one of its minors can be computed in PSPACE. Designing an improved algorithm to compute the multivariate resultant in $\mathsf{FP}^{\mathsf{CH}}$ will be the main step in our proof that $\mathsf{HN} \in \mathsf{CH}$.

1.3.2 The counting hierarchy

Before describing how to compute the resultant in $\mathsf{FP}^{\mathsf{CH}}$, let us see what sort of algorithmic power CH provides. Just as the polynomial-time hierarchy PH is connected to bounded-depth boolean circuits built from AND, OR, and NOT gates [FSS84], the counting hierarchy is related to bounded-depth boolean circuits built from threshold and negation gates. For us, it will be useful to view CH through its characterization in terms of polynomially-bounded majority quantifiers: a language $L \subseteq \{0,1\}^*$ is in CH if there is a polynomially-bounded function p and a polynomial-time Turing machine M such that

$$x \in L \iff \operatorname{Maj} y_1 \in \{0,1\}^{p(|x|)} \cdots \operatorname{Maj} y_k \in \{0,1\}^{p(|x|)} M(x, y_1, \ldots, y_k) \text{ accepts},$$

where the majority quantifier $\operatorname{Maj} y$ stipulates that the subsequent formula is true for the majority of choices of the variable y. With a single majority quantifier, we can compute the majority function (and more generally, any threshold function) over an exponentially-large set of bits, as long as any single bit in this set can be computed in polynomial time. By repeating this observation, we can evaluate an exponentially-large constant-depth threshold circuit in CH, as long as the connectivity properties of this circuit can be decided by a polynomial-time Turing machine. More precisely, if a boolean function $f : \{0,1\}^* \to \{0,1\}$ can be computed by a polylogtime-uniform family of threshold circuits of constant depth and exponential size, then $f \in \mathsf{CH}$.

Bounded-depth threshold circuits are surprisingly powerful. Basic operations on rational numbers, such as iterated addition, iterated multiplication, and division with remainder, can all be computed by logtime-uniform families of TC^0 circuits [HAM02]. This implies that threshold circuits can simulate arithmetic circuits while only losing a constant factor in depth and a polynomial factor in size. In particular, in CH, we can simulate a polylogtime-uniform family of arithmetic circuits of bounded depth and exponential size. This is the main property of CH that we use in our algorithms.

1.3.3 Computing the resultant in constant depth

Now we return to the task of computing the multivariate resultant in $\mathsf{FP}^{\mathsf{CH}}$. From our discussion on the power of CH, it suffices to construct a polylogtime-uniform family of arithmetic circuits of constant depth and exponential size that computes the resultant, and this is the route we will take.

Just as in the case of two polynomials, the multivariate resultant can be expressed in terms of the evaluations of one polynomial at the common roots of the others. To set notation, let

$$\overline{F}_i(\mathbf{x}) \coloneqq F_i(x_0, \ldots, x_{n-1}, 0)$$
$$f_i(\mathbf{x}) \coloneqq f_i(x_0, \ldots, x_{n-1}, 1).$$

[4] Because the polynomials $F_0, \ldots, F_n$ are homogeneous, the all-zeroes point is always a solution of the system $F_0 = \cdots = F_m = 0$. The resultant detects when this system has a nontrivial solution, or equivalently, when this system has a solution in projective space $\mathbb{P}^n$.

Then, assuming the smaller resultant $\mathrm{Res}(\overline{F}_0, \dots, \overline{F}_{n-1})$ is not zero, the *Poisson formula* shows the resultant factors as

$$\mathrm{Res}(F_0, \dots, F_n) = \mathrm{Res}(\overline{F}_0, \dots, \overline{F}_{n-1})^{d_{n-1}} \prod_{\boldsymbol{\alpha} \in \mathrm{V}(f_0, \dots, f_{n-1})} f_n(\boldsymbol{\alpha})^{m(\boldsymbol{\alpha})},$$

where $\mathrm{V}(f_0, \dots, f_{n-1}) \subseteq \mathbb{C}^n$ is the set of common roots of $f_0, \dots, f_{n-1}$ (which is necessarily finite by the assumption on the smaller resultant), and $m(\boldsymbol{\alpha})$ is the multiplicity of $\boldsymbol{\alpha}$ as a solution to the system $f_0 = \cdots = f_{n-1} = 0$. By recursively applying the Poisson formula to the smaller resultant $\mathrm{Res}(\overline{F}_0, \dots, \overline{F}_{n-1})$, we see that the original resultant $\mathrm{Res}(F_0, \dots, F_n)$ we wanted to compute can be expressed as a product of terms of the form $\prod_{\boldsymbol{\alpha} \in \mathrm{V}(f_0, \dots, f_{n-1})} f_n(\boldsymbol{\alpha})^{m(\boldsymbol{\alpha})}$. If we can compute one such a term using a bounded-depth arithmetic circuit, then by computing all terms in parallel and multiplying them, we obtain a bounded-depth arithmetic circuit for the resultant itself.

To compute a product of the form $\prod_{\boldsymbol{\alpha} \in \mathrm{V}(f_0, \dots, f_{n-1})} f_n(\boldsymbol{\alpha})^{m(\boldsymbol{\alpha})}$, it appears that we need to solve the system of equations $f_0 = \cdots = f_{n-1} = 0$. In the case $n = 1$, this can avoided by using the Girard–Newton identities [AW24]. While there are similar identities in the case $n \geq 2$ (see [AiK81] and references therein), they are more complicated, and it is not clear if they can be implemented in constant depth. Instead, we will use a computationally explicit version of the implicit function theorem to express the solutions of $f_0 = \cdots = f_{n-1} = 0$ as power series in the coefficients of $f_0, \dots, f_{n-1}$, where initial segments of these power series can be computed by constant-depth circuits. A similar idea appears in recent work of Bhattacharjee, Kumar, Rai, Ramanathan, Saptharishi, and Saraf [BKR+25a], who used Lagrange inversion to show that low-depth arithmetic circuits are closed under factorization. Their work, particularly its use of an explicit version of the implicit function theorem, was a key inspiration for the results of this paper.

The implicit function theorem requires the Jacobian of $(f_0, \dots, f_{n-1})$ to be invertible at $\boldsymbol{\alpha}$, which may not be true of the system we are given. To remedy this, we use the method of homotopy continuation to obtain the roots of $f_0 = \cdots = f_{n-1} = 0$ by solving a different—but related—system of equations. We will choose another system of homogeneous equations $G_0, \dots, G_n \in \mathbb{C}[x_0, \dots, x_n]$, unrelated to the F_i, where the solutions of $G_0 = \cdots = G_n = 0$ are explicitly known in advance and at which the Jacobian is invertible. Letting t be a fresh variable, we then consider the system of equations $H_0 = \cdots = H_n = 0$, where H_i is given by

$$H_i(t, \mathbf{x}) \coloneqq (1 - t) \cdot G_i(\mathbf{x}) + t \cdot F_i(\mathbf{x}).$$

This system is easy to solve at $t = 0$, since it simplifies to $G_0 = \cdots = G_n = 0$, whose solutions are known to us. At $t = 1$, we recover the original system we wanted to solve. In particular, if we can instead compute the resultant $\mathrm{Res}(H_0, \dots, H_n)$, then evaluating at $t = 1$ will recover $\mathrm{Res}(F_0, \dots, F_n)$.

Let

$$g_i(\mathbf{x}) \coloneqq G_i(x_0, \dots, x_{n-1}, 1)$$
$$h_i(t, \mathbf{x}) \coloneqq H(t, x_0, \dots, x_{n-1}, 1).$$

For each solution $\boldsymbol{\alpha} \in \mathbb{C}^n$ of the system $g_0 = \cdots = g_{n-1} = 0$, there is a corresponding power series solution $\boldsymbol{\varphi}_{\boldsymbol{\alpha}}(t) \in \mathbb{C}[\![t]\!]^n$ of $h_0 = \cdots = h_{n-1} = 0$, and all solutions of $h_0 = \cdots = h_{n-1} = 0$ arise in this way. The coefficients of the power series solution $\boldsymbol{\varphi}_{\boldsymbol{\alpha}}(t)$ are polynomial functions of the coefficients of the original system $f_0, \dots, f_{n-1}$. Importantly, the first $(n+1)d^n$ coefficients of these power series can be computed from the coefficients of $f_0, \dots, f_{n-1}$ using an arithmetic circuit of constant depth and size bounded by $d^{\mathrm{poly}(n)}$, where $d = \max(d_0, \dots, d_{n-1})$. This allows us to

compute an approximate solution $\widetilde{\boldsymbol{\varphi}}_{\boldsymbol{\alpha}}(t) \in \mathbb{C}[t]^n$ that agrees with $\boldsymbol{\varphi}_{\boldsymbol{\alpha}}(t)$ up to degree $(n+1)d^n$. By evaluating h_n on each of the approximate solutions $\widetilde{\boldsymbol{\varphi}}_{\boldsymbol{\alpha}}(t)$ and using the Poisson formula, we compute a polynomial $\widetilde{\mathrm{Res}}(H_0, \ldots, H_n)$ that agrees with the resultant $\mathrm{Res}(H_0, \ldots, H_n)$ modulo $t^{(n+1)d^n}$. Because the resultant is a polynomial of degree at most $(n+1)d^n$, we can use interpolation to recover the terms of $\widetilde{\mathrm{Res}}(H_0, \ldots, H_n)$ of degree at most $(n+1)d^n$ in t, which necessarily equal the resultant $\mathrm{Res}(H_0, \ldots, H_n)$.

This approach produces a family of arithmetic circuits of constant depth and exponential size to compute the resultant. To conclude that the resultant can be computed in $\mathsf{FP}^{\mathsf{CH}}$, we need to ensure that this family of circuits is sufficiently uniform. Most parts of this algorithm are easily shown to be uniform. The only part that is not immediately uniform is our use of evaluation-interpolation to compute the coefficients of a polynomial from its evaluations. Doing this requires constructing a uniform family of constant-depth arithmetic circuits to compute the inverse of a Vandermonde matrix, which we do using an explicit description of the inverse of the Vandermonde matrix at the points $1, 2, \ldots, N$ in terms of Stirling numbers.

So far, we have seen how to compute the resultant of polynomials over $\mathbb{Q}$ in $\mathsf{FP}^{\mathsf{CH}}$. The same algorithm would likewise produce constant-depth arithmetic circuits to compute the resultant over all fields, but the notion of uniformity for arithmetic circuits becomes more difficult to work with over finite fields. In particular, our circuits would need access to constants from an extension field, and one must bound the uniformity of these constants in some way. In contrast, over $\mathbb{Q}$, constant-free circuits are sufficient for our purposes, and one only needs to bound the uniformity of the circuit's structure. To compute the resultant over other fields, such as the finite field $\mathbb{F}_p$, we use the fact that the resultant is essentially the same polynomial over all fields. In particular, for polynomials $F_0, \ldots, F_n \in \mathbb{F}_p[x_0, \ldots, x_n]$, if we lift F_i to the polynomial $\hat{F}_i \in \mathbb{Z}[x_0, \ldots, x_n]$, then the resultant satisfies

$$\mathrm{Res}(F_0, \ldots, F_n) = \mathrm{Res}(\hat{F}_0, \ldots, \hat{F}_n) \bmod p.$$

Thus, to compute the resultant over $\mathbb{F}_p$ in $\mathsf{FP}^{\mathsf{CH}}$, we lift the input to the integers, compute the resultant over the integers, and then reduce this value modulo p.

1.3.4 From homogeneous to affine systems

Computing the resultant in $\mathsf{FP}^{\mathsf{CH}}$ allows us to decide if a system of n homogeneous equations in n variables has a nonzero solution, or in other words, if it has a solution in projective space $\mathbb{P}^{n-1}$. It is not difficult to extend this to an algorithm that handles non-square projective systems: if we are given m equations $F_1 = \cdots = F_m = 0$ in n variables where $m > n$, one can show that the system $G_1 = \cdots = G_n = 0$ obtained by taking each G_i to be a random linear combination of the F_j has the same set of solutions with high probability. Deciding the solvability of inhomogeneous equations is more difficult. The natural approach is to homogenize a given inhomogeneous system, but this does not necessarily preserve the solvability of the system of equations. For example, the system $x + y - 1 = x + y - 2 = 0$ has no solution, but its homogenization $x + y - z = x + y - 2z = 0$ has the nonzero solution $(1, -1, 0)$.

Fortunately, this problem has already been solved. The *generalized characteristic polynomial* of Canny [Can90], later extended by Ierardi [Ier89a], allows us to decide if an inhomogeneous system of equations has a solution by performing a resultant computation on a slight perturbation of the given system. A small variation on the construction of the generalized characteristic polynomial produces a univariate polynomial whose roots are in one-to-one correspondence with solutions of the given system of equations, assuming this number is finite. This reduces the task of counting solutions to a multivariate system of equations to counting the number of distinct roots of a univariate

polynomial of exponentially-large degree, but whose coefficients can be computed in $\mathsf{FP}^{\mathsf{CH}}$. To do this in $\mathsf{FP}^{\mathsf{CH}}$, we adapt the squarefree factorization algorithm of Andrews and Wigderson [AW24] that computed the squarefree decomposition of a univariate polynomial using arithmetic circuits of constant depth and polynomial size. Scaled up to polynomials of exponential degree, this results in an $\mathsf{FP}^{\mathsf{CH}}$ algorithm that counts the number of distinct roots of a given polynomial.

1.4 Organization

The rest of this paper is organized as follows. Section 2 introduces notation and also introduces the notion of uniformity we use for boolean and arithmetic circuit families. It also covers preliminary material in computational algebra. In Section 3, we show that basic operations, including polynomial interpolation, can be performed by uniform families of constant-depth arithmetic circuits. Section 4 introduces the multivariate resultant, collecting well-known properties of the resultant and describing how the resultant can be used to decide the Nullstellensatz and count solutions to zero-dimensional systems of equations. In Section 5, we construct a uniform family of constant-depth arithmetic circuits that compute the multivariate resultant. Section 6 transfers this circuit family to the boolean setting, obtaining CH algorithms for the multivariate resultant and, as a consequence, the Nullstellensatz. Finally, we conclude in Section 7 with a few straightforward applications of our algorithm for the Nullstellensatz.

2 Preliminaries

2.1 Notation and conventions

Throughout this work, we use $\mathbb{F}$ to denote a field. Various results that we use and present have differing requirements on the field $\mathbb{F}$ (for example, some results require the field to be large enough). Each statement will clearly state the requirement on the field $\mathbb{F}$. If we make no such qualification in the statement of a particular result, then that result holds for any field. We denote by $\overline{\mathbb{F}}$ the algebraic closure of $\mathbb{F}$.

The *coefficient subfield* of a field $\mathbb{F}$ is the largest algebraic extension of the prime field of $\mathbb{F}$ within $\mathbb{F}$. For example, if $\mathbb{F} = \mathbb{Q}(y)$, then the coefficient subfield of $\mathbb{F}$ is $\mathbb{Q}$, and if $\mathbb{F} = \mathbb{F}_{p^a}$, then $\mathbb{F}$ is its own coefficient subfield.

We write $\mathbb{F}[x_1, \ldots, x_n]$ and $\mathbb{F}(x_1, \ldots, x_n)$, respectively, for the ring of polynomials and field of rational functions in the variables $x_1, \ldots, x_n$ and with coefficients in $\mathbb{F}$. We abbreviate vectors as $\mathbf{x} = (x_1, \ldots, x_n)$. For $\mathbf{a} \in \mathbb{N}^n$, we write $\mathbf{x}^{\mathbf{a}}$ for the monomial $x_1^{a_1} \cdots x_n^{a_n}$. We write $|\mathbf{a}|$ for the sum $\sum_{i=1}^n a_i$.

By a *form* of degree d, we mean a homogeneous element of $\mathbb{F}[x_1, \ldots, x_n]$ of degree d. We denote forms by capital letters and inhomogeneous polynomials by lowercase letters. Often, we will work with a pair of polynomials where one is the (de-)homogenization of the other. In these cases, we use upper- and lowercase variants of the same letter as a mnemonic device to indicate this relationship. For example, if $F \in \mathbb{F}[x_0, x_1, \ldots, x_n]$ is a form, we will denote its dehomogenization $F(1, x_1, \ldots, x_n)$ by $f(x_1, \ldots, x_n)$. Likewise, if $f \in \mathbb{F}[x_1, \ldots, x_n]$ is an inhomogeneous polynomial of degree d, we write F for its homogenization $x_0^d f(x_1/x_0, \ldots, x_n/x_0)$. In some settings we might choose to homogenize and dehomogenize using a different variable, which will be made clear in context.

For polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, we write $\mathrm{V}(f_1, \ldots, f_m)$ for the set of common zeroes of $f_1, \ldots, f_m$ in $\overline{\mathbb{F}}^n$. When dealing with forms $F_1, \ldots, F_m \in \mathbb{F}[x_0, \ldots, x_n]$, we instead consider their zero set $\mathrm{V}(F_1, \ldots, F_m)$ in the projective space $\mathbb{P}^n_{\overline{\mathbb{F}}}$. We usually suppress the field dependence from this notation and just write $\mathbb{P}^n$ for n-dimensional projective space.

We use $\mathrm{Vand}(x_1, \ldots, x_n)$ to denote the Vandermonde matrix defined using $x_1, \ldots, x_n$. Specifically, the entry in position i, j in the above matrix is x_i^{j-1}.

We will frequently have to refer to the trailing term and trailing part of a polynomial with respect to some variable. For convenience, we define the following piece of notation.

Definition 2.1. Suppose $f \in \mathbb{F}[t, y_1, \ldots, y_m]$ is a non-zero polynomial. The *trailing term of f with respect to t*, denoted by $\mathrm{TT}_t\, f(y_1, \ldots, y_m)$, is the coefficient of the lowest-degree term in f when viewed as a polynomial in t. Concretely, if $f = t^i f_i(y_1, \ldots, y_m) + t^{i+1} f_{i+1}(y_1, \ldots, y_m) + \cdots + t^d f_d(y_1, \ldots, y_m)$ where $f_i(\mathbf{y}) \neq 0$, then $\mathrm{TT}_t\, f = f_i$.

The *trailing part of f with respect to t*, written $\mathrm{TP}_t\, f$, is the polynomial obtained by factoring out the highest power of t from f. In the above setting, we have $\mathrm{TP}_t\, f = f_i + t f_{i+1} + \cdots + t^{d-i} f_d$.

By convention, the trailing term and the trailing part of the zero polynomial are zero. ◊

We follow the convention that a variety is any set of solutions of a system of polynomial equations over an algebraically closed field. In particular, varieties need not be irreducible. We also follow the convention that the degree of a reducible projective or affine variety (denoted $\deg V$) is the sum of the degrees of all irreducible components. The greatest common divisor of any two univariate polynomials with coefficients in a field $\mathbb{F}$ will always be monic.

2.2 Uniformity of boolean and arithmetic circuit families

In this subsection, we discuss the notion of uniformity for boolean and arithmetic circuit families, starting with the boolean case. We restrict our discussion to families of boolean threshold circuits.

There are a number of different ways of measuring the uniformity of boolean circuits [Vol99]. In our work, we measure the uniformity of a boolean circuit by the complexity of its *direct connection language*, defined below. This notion is particularly well-suited for very weak circuit classes, such as constant-depth circuits. We follow the presentation of Vollmer [Vol99], but with one important modification: while most work on uniformity addresses circuit families indexed by a single parameter $n \in \mathbb{N}$, we will work with circuit families indexed by an arbitrary number of parameters $n_1, \ldots, n_k \in \mathbb{N}$, and so we adapt the notion of the direct connection language to this setting. Before we define the direct connection language, we must define the notion of *admissible encodings* of circuits.

Definition 2.2 ([Vol99, Definition 2.14])**.** Let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of boolean circuits of size $s := s(k, n_1, \ldots, n_k)$. An *admissible encoding of $\mathcal{C}$* is a numbering of the gates of each circuit $C = C_{k,n_1,\ldots,n_k}$ in the family with the following properties.

1. If C has m input gates, then they are numbered $0, \ldots, m-1$.
2. If C has m' output gates, then they are numbered $m, \ldots, m+m'-1$.
3. There is a constant c, depending only on the circuit family, such that the binary representation of the number of any gate in C has length at most $c \cdot \log(s)$. ◊

For circuit families of threshold circuits with admissible encodings, we define the direct connection language as follows.

Definition 2.3 (Direct connection language)**.** Let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of threshold circuits with an admissible encoding. Suppose a numbering of the gate types (AND, OR, NOT, MAJ, 0, 1) is fixed. The *direct connection language of $\mathcal{C}$*, denoted by $L_{DC}(\mathcal{C})$, is the language consisting of binary encodings of tuples $(k, n_1, \ldots, n_k, a, p, b)$, where

1. a is the number of a gate in $C_{k,n_1,\ldots,n_k}$.
2. If $p \neq \epsilon$, then b is the number of a gate in $C_{k,n_1,\ldots,n_k}$ that is a direct predecessor of a.

3. If $p = \epsilon$, then b encodes the gate type of a (using the fixed numbering of gate types). $\Diamond$

By modifying the allowed gate types, the above definition can also be extended to other types of families of boolean circuits, however we will only work with threshold circuits in this article. The uniformity of a circuit family can be measured using the complexity of the direct connection language as follows.

Definition 2.4. Let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of threshold circuits of size $s := s(k, n_1, \ldots, n_k)$ with an admissible encoding. We say that the circuit family $\mathcal{C}$ is *polylogtime-uniform* if

1. its direct connection language $L_{DC}(\mathcal{C})$ can be decided in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$, and
2. the number of inputs and outputs of $C_{k,n_1,\ldots,n_k}$, and an upper bound on the size s can be computed in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$, given $(k, n_1, \ldots, n_k)$ as input.

We say that the circuit family $\mathcal{C}$ is *logtime-uniform* if both of these tasks can be done in time linear in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$. $\Diamond$

For circuit families indexed by a single parameter, the above definition requires that a Turing machine can decide the direct connection language in time polylogarithmic in the size of the circuit and the index within the circuit family. We remark that the second item above is not usually part of the standard definition. We include it to make it easier to compose circuits uniformly.

In the preceding definitions, the first index is the count of the number of remaining indices. In the circuits we design, we will extend this slightly, and allow the number of indices to be a function of the first index. This will always be a very simple function, in all applications the number of indices will be linear in the first index, for example twice the first index plus two. In the cases where the number of indices is just constant, for example families indexed by a single parameter, we do not use the first index as the count of the number of indices. For example, a single indexed family of circuits is just written $(C_n)_n$, as opposed to $(C_{1,n})_n$ as the above notation might suggest. We also use vector notation as short hand for multi-indexed families. For example, we use $(C_{\mathbf{n}})_{\mathbf{n}}$ to denote a family indexed by $k, n_1, \ldots, n_k$, and use $C_{\mathbf{n}}$ to denote a specific circuit in the family. In all cases, the exact number of indices will always be clear from context.

Remark 2.5. The definition of admissible encodings gives us a lot of freedom when picking how the gates have to be numbered. Other than the input and output gates, we are allowed to number the gates in any way, as long as the restriction on the length is respected. This freedom allows us to number our gates using more than just the integers 1 through s: it makes it possible to interpret tuples of integers as gate names, as we explain now. In what follows, to avoid confusion, when we number gates by objects other than natural numbers, we will refer to the *name* of a gate instead of its number.

Here is for instance how we can design circuits where gate names are pairs of natural numbers. There exist efficient schemes to encode pairs of natural numbers as a single natural number, and these encoding schemes are efficiently invertible when an inverse exists. If we fix such a scheme, we can use pairs such as (i, j) as names, with the understanding that the actual number of the gate is encoding of the pair (i, j) as a single natural number.

Care must be taken to respect the condition on the input and output numbering: we have to ensure that the encoding scheme is such that these gates are mapped to $0, \ldots, m-1$ and $m, \ldots, m + m' - 1$. In all instances where we rely on such naming conventions, given the name of an input, resp. output, gate (by means of a pair, or tuple, of integers) we will always be able to

decide that this is an input, resp. output, and determine its index in the allotted time; the same will hold for the converse direction. For gates other than inputs and outputs, we will use a scheme as described above to map a pair (or tuple) of integers to a single integer greater that $m + m' - 1$ and conversely (recall that the number of inputs and outputs can be computed in polylogarithmic time). ◊

Remark 2.6. Suppose we are working with a polylogtime-uniform circuit family. The gate name and the index of the circuit can be used to efficiently determine the type of a gate as follows. Given the index $(k, n_1, \ldots, n_k)$ of a circuit and the name a of a gate in the circuit, we iterate over all gate types b and check if the tuple $(k, n_1, \ldots, n_k, a, \epsilon, b)$ is in the direct connection language. Since there are a constant number of gate types, this allows us to compute the type of a gate, given its name.

On input $(k, n_1, \ldots, n_k, a)$, we can also decide if a candidate gate name a is valid in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$. We first compute an upper bound s' on s; this can be done in the allotted time, by definition; then, we test if the length of a is at most $c \log(s')$, for the constant c from Definition 2.2. Since $\log(s')$ itself is polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$, this also fits in our time bound. If not, we reject. Else, we proceed as above, forming the strings $(k, n_1, \ldots, n_k, a, \epsilon, b)$ for all gate types b and checking if they belong to the direct connection language.

Similarly, we can always determine if a given gate is an input or output gate, since we can determine the number of inputs and outputs to a particular circuit. ◊

Remark 2.7. In our circuits, we will also carefully order the inputs and outputs to make other computations easier. For example, if the output of a circuit represents the entries of a matrix, we might choose to arrange the outputs in row major order. If the outputs of the circuits represent coefficients of a polynomial, we will fix a monomial ordering and order the outputs accordingly. ◊

We now define uniform families of arithmetic circuits. We start with the definition of an arithmetic circuit.

Definition 2.8. Let $\mathbb{F}$ be a field and let $\mathbb{F}(\mathbf{x})$ be the field of rational functions in the variables $x_1, \ldots, x_n$ over $\mathbb{F}$. An *arithmetic circuit over* $\mathbb{F}$ is a directed acyclic graph. The vertices of this graph of in-degree zero are either called *input gates* and are labeled by a variable x_i, or called *constant gates* and are labeled by a field element $\alpha \in \mathbb{F}$. Vertices of positive in-degree are called *internal gates* and are labeled by an operation from $\{+, \times, \div\}$. Vertices of out-degree zero are called *output gates*. Each gate of the circuit naturally computes an element of $\mathbb{F}(\mathbf{x})$, assuming no division by zero takes place in the circuit, which we require. If $\{f_1, \ldots, f_m\}$ are the functions computed by the output gates of the circuit, we say that the circuit *computes* the functions $f_1, \ldots, f_m$. The *size* of the circuit is the number of wires in the circuit. The *depth* of the circuit is the length of the longest path from an input to an output of the circuit.

If the only constants labeling the input gates of the circuit are $0, +1$, and -1, we say that the circuit is *constant-free*. For a fixed variable x_i, if the subtree rooted at the denominator of each division gate in the circuit does not contain the input gate corresponding to x_i, then we say that the circuit is *division-free with respect to* x_i. If X' is a subset of variables and the circuit is division-free with respect to every variable in X', we say the circuit is *division-free with respect to* X'. If the circuit is division-free with respect to all variables, we say the circuit is *weakly division-free*. ◊

When working with families of arithmetic circuits, uniformity also has to take into account the complexity of constructing the field elements used in the circuit (the elements α in Definition 2.8). Various definitions of uniformity that take the field elements into account are discussed in [Gat86]. In this work, we will restrict ourselves to circuits that are constant-free. Any field element that we want to use within the circuit has to be constructed using $+1$ and -1. This sidesteps the issue of

having to define a notion of uniformity for the field elements used in the circuit family. We can thus work with a notion of uniformity for arithmetic circuit families that closely mirrors the notion for boolean circuit families.

We will later want to simulate our arithmetic circuits using boolean threshold circuits. We will define our notion of uniformity in a way that affords this simulation. In the boolean case, the direct connection language only answered queries of the form *is a gate of type* b*?* and *is gate* b *a predecessor of gate* a*?*. For the sake of the simulation, we place a slightly more stringent requirement on the uniformity of arithmetic circuit families. Given a gate a with k predecessors, we want to be able to answer queries of the form *is gate* b *the* i^{th} *predecessor of gate* a*?*. For this, we of course require an ordering of all predecessors of each gate in the circuit. This will be part of the definition of an admissible encoding of an arithmetic circuit.

Definition 2.9 (Admissible encoding)**.** Let $\mathbb{F}$ be a field and let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of constant-free arithmetic circuits over $\mathbb{F}$ of size $s \coloneqq s(k, n_1, \ldots, n_k)$. An *admissible encoding of* $\mathcal{C}$ is a numbering of the gates of each circuit $C = C_{k,n_1,\ldots,n_k}$ in the family with the following properties.

1. If C has m input gates, then they are numbered $0, \ldots, m-1$.
2. If C has m' output gates, then they are numbered $m, \ldots, m+m'-1$.
3. There is a constant c, depending only on the circuit family, such that the binary representation of the number of any gate in C has length at most $c \cdot \log(s)$.

Further, for each gate $a \in C$, we fix a numbering of the set of immediate predecessors of a in C. We require that this numbering of predecessors is contiguous, that is, it starts at 1 and ends at the arity of a. We do not require the numbering of predecessors to respect the numbering of the gates themselves: if b_1 and b_2 are two predecessors of a, with b_1 preceding b_2 in the numbering of the gates of C, we do not require b_1 to precede b_2 in the numbering of the predecessors of a. For division gates, we require that the numerator occurs before the denominator in the numbering of the predecessors. ◊

Definition 2.10. Let $\mathbb{F}$ be a field and let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of constant-free arithmetic circuits over $\mathbb{F}$ with an admissible encoding. Suppose a numbering of the gate types $(+, -, \times, \div, +1, 0, -1)$ is fixed. The *direct connection language of* $\mathcal{C}$, denoted by $L_{DC}(\mathcal{C})$, is the language consisting of binary encodings of tuples $(k, n_1, \ldots, n_k, a, p, b)$, where

1. a is the number of a gate in $C_{k,n_1,\ldots,n_k}$.
2. If $p \neq \epsilon$, then b is the number of the p^{th} predecessor gate in $C_{k,n_1,\ldots,n_k}$ of a.
3. If $p = \epsilon$, then b encodes the gate type of a (using the fixed numbering of gate types). ◊

As in the boolean case, we measure the uniformity of a circuit family $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ by the complexity of its direct connection language $L_{DC}(\mathcal{C})$.

Definition 2.11. Let $\mathbb{F}$ be a field and let $\mathcal{C} = (C_{k,n_1,\ldots,n_k})_{k,n_1,\ldots,n_k \in \mathbb{N}}$ be a family of constant-free arithmetic circuits of size $s \coloneqq s(k, n_1, \ldots, n_k)$ over $\mathbb{F}$ with an admissible encoding. We say that the circuit family $\mathcal{C}$ is *polylogtime-uniform* if

1. its direct connection language $L_{DC}(\mathcal{C})$ can be decided in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$,
2. the number of inputs and outputs of $C_{k,n_1,\ldots,n_k}$, and an upper bound on the size s can be computed in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$, given $(k, n_1, \ldots, n_k)$ as input.

3. the arity of a gate can be computed in time polynomial in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$, given $(k, n_1, \ldots, n_k)$ and the gate name as input.

We say that the circuit family $\mathcal{C}$ is *logtime-uniform* if these tasks can be carried out in time linear in $\log(s) + \log(k) + \sum_{i=1}^{k} \log(n_i)$. ◊

Remarks 2.5 to 2.7 apply verbatim in the arithmetic setting.

2.3 Integer, rational, and finite field arithmetic

In this section, we discuss how elements of various domains are represented by Turing machines and boolean circuits. We also outline some basic integer and rational arithmetic operations that are in TC^0.

Elements of $\mathbb{Z}$ will be represented in base two with an additional sign bit. For an integer $a \in \mathbb{Z}$, the *height* of a is defined to be $\lceil \log_2(|a|) \rceil$, therefore, an integer of height h is represented using $h+1$ bits (taking into account the sign bit). The height of a polynomial with coefficients in $\mathbb{Z}$ is defined to be the maximum of the heights of its coefficients. In this representation, the following holds.

Theorem 2.12. *The following functions are in logtime-uniform* TC^0.

1. Iterated Integer Addition: *Given n integers $a_1, \ldots, a_n$ as input, each consisting of at most n bits, compute the sum $a_1 + \cdots + a_n$.*

2. Iterated Integer Multiplication: *Given n integers $a_1, \ldots, a_n$ as input, each consisting of at most n bits, compute the product $a_1 \cdots a_n$.*

3. Integer Division: *Given integers a and b as input, compute $\lfloor a/b \rfloor$.*

4. Polynomial Division with Remainder: *given integers $a_0, \ldots, a_n$ and $b_0, \ldots, b_m$ as input, where $b_m \neq 0$ and $n \geq m$, compute $d \coloneqq b_m^{n-m+1}$ and integers $q_0, \ldots, q_{n-m}, r_0, \ldots, r_{m-1}$ such that for*

$$f(x) \coloneqq a_n x^n + \cdots + a_1 x + a_0 \qquad q(x) \coloneqq q_{n-m} x^{n-m} + \cdots + q_1 x + q_0$$
$$g(x) \coloneqq b_m x^m + \cdots + b_1 x + b_0 \qquad r(x) \coloneqq r_{m-1} x^{m-1} + \cdots + r_1 x + r_0,$$

   we have $d \cdot f(x) = q(x)g(x) + r(x)$.

5. Iterated Polynomial Multiplication: *given integers $a_{1,0}, \ldots, a_{m,n}$ as input, compute integers $b_0, \ldots, b_{mn}$ such that*

$$b_{mn} x^{mn} + b_{mn-1} x^{mn-1} + \cdots + b_1 x + b_0 = \prod_{i=1}^{m} (a_{i,n} x^n + a_{i,n-1} x^{n-1} + \cdots + a_{i,1} x + a_{i,0}).$$

Proof. Item 1 is classical, see [RT92]. Items 2 and 3 are the main result of [HAM02]. Items 4 and 5 are from [HAM02, Corollary 6.5]. For all items above, the number of input and output bits are easily computable from the index of a circuit in the corresponding family. □

The operation performed in item 4 is *pseudodivision*. The polynomials q and r in the notation above are called the *pseudoquotient* and *pseudoremainder* of f and g respectively.

Multivariate polynomials will be represented in the dense representation. In other words, the coefficient of every monomial up to a specified degree, including the ones that are zero, will be

listed. To multiply polynomials in $\mathbb{Z}[y_1, \ldots, y_k]$, we use the Kronecker substitution to reduce to the univariate case. If we want to multiply m such polynomials of degree d, then we substitute $z^{(md+1)^i}$ for each y_i. This gives us m polynomials of degree $(md)^{O(k)}$, which we can multiply using the univariate multiplication circuit provided by Theorem 2.12. The coefficients of the $\mathbf{y}$ monomials in the product can be read off from the coefficients of z in the product after the above substitution.

An element of $q \in \mathbb{Q}$ will be represented by a pair of integers a, b with $b \neq 0$ such that $q = a/b$, where a, b may have common factors. The reason we allow a, b to have common factors is that computing integer GCD is not known to be in TC^0, therefore the circuits we are working with cannot convert a fraction to lowest terms. The height of a pair of integers a, b is simply the maximum of the heights of a and b.

Iterated multiplication and addition of rational numbers in the above representation can be carried out by polylogtime-uniform TC^0 circuits as follows. For iterated multiplication, we separately multiply the numerators and denominators using the circuits for iterated integer multiplication (Theorem 2.12). This directly gives us a representation of the product. For iterated addition, we first bring all elements to a common denominator. This common denominator is the product of the denominators of the input, and so a representation of each input over this common denominator can be computed using the circuit for iterated integer multiplication from Theorem 2.12. Once we have this representation, we can add the numerators using the circuit for iterated integer addition, again from Theorem 2.12.

By the primitive element theorem, every number field $\mathbb{K}$ is of the form $\mathbb{Q}[\alpha] \cong \mathbb{Q}[z]/(g(z))$, where α is an algebraic number with minimal polynomial $g(z)$. Elements of a number field $\mathbb{K}$ will therefore be represented by polynomials in $\mathbb{Q}[z]$ of degree less than $\deg g$. Whenever we work with such a number field, we will assume that $g(z)$ is given as part of the input.

Elements of $\mathbb{F}_p$, where p is a prime, will be represented by an integer between 0 and $p-1$ inclusive. The *height* of an element of $\mathbb{F}_p$ is defined to be $\log(p)$.

The finite field $\mathbb{F}_{p^a}$ is isomorphic to $\mathbb{F}_p[z]/(g(z))$ for some irreducible polynomial $g(z) \in \mathbb{F}_p[z]$ of degree a. Elements of the field $\mathbb{F}_{p^a}$ will therefore be represented by polynomials in $\mathbb{F}_p[z]$ of degree less than a. Whenever we work with such a finite field, we will assume that $g(z)$ is given as part of the input and that $g(z)$ is monic. The *height* of an element of $\mathbb{F}_{p^a}$ is defined to be $a \log(p)$. For each of the rings R discussed above, the height of a polynomial with coefficients in R is the maximum of the heights of the coefficients.

Our algorithms will occasionally require the fields we work with to have coefficient subfields that are larger than some bound B. If our inputs lie in a field with coefficient subfield $\mathbb{F}_{p^a}$ where p^a is less than B, then we will instead pass to a field extension that is large enough and work over that field instead. The following lemma shows that this can be done efficiently by a Turing machine.

Lemma 2.13. *Given a finite field $\mathbb{F}_{p^a}$ via an irreducible polynomial $g(y) \in \mathbb{F}_p[y]$ of degree a, and a bound B, there is a Las Vegas algorithm that explicitly constructs a finite field $\mathbb{F}_{p^b}$ such that $p^b \geq B$ and $\mathbb{F}_{p^a}$ is a subfield of $\mathbb{F}_{p^b}$. The algorithm also constructs an $\mathbb{F}_p$-linear map ϕ (represented as a matrix in the powers of y bases) that maps $\mathbb{F}_{p^a}$ to an isomorphic subfield of $\mathbb{F}_{p^b}$. The expected running time of the algorithm is polynomial in $a \log p$ and $\log |B|$.*

Proof. Recall that $\mathbb{F}_{p^n}$ is a subfield of $\mathbb{F}_{p^m}$ if and only if m is a multiple of n. We therefore pick b to be the first multiple of a that is larger than $\log |B| / \log p$. To construct an irreducible polynomial of degree b, we simply sample a random polynomial of degree b in $\mathbb{F}_p[y]$ and test for irreducibility. By [GG13, Corollary 14.39], this can be performed using $\widetilde{O}\big(b^3 + b^2 \log p\big)$ operations in $\mathbb{F}_p$ in expectation. Let this irreducible polynomial be $h(y)$.

To construct the map between $\mathbb{F}_{p^a}$ and $\mathbb{F}_{p^b}$, we simply find a root of g in $\mathbb{F}_{p^b}$. That such a root exists follows from Fermat's little theorem and the structure of the factorization of polynomials

in finite fields, see for example [GG13, Theorem 14.2]. By [GG13, Corollary 14.16], finding this root requires $\widetilde{O}(ab \log p)$ operations in $\mathbb{F}_{p^b}$ in expectation. If α is such a root, then the map $\phi : \mathbb{F}_p[y]/(g(y)) \to \mathbb{F}_{p^b}$ sending y to α is an isomorphism between $\mathbb{F}_{p^a}$ and a subfield of $\mathbb{F}_{p^b}$. □

We remark that the method above is far from the optimal way of constructing a compatible field extension. However, it will suffice for our applications.

3 Uniformity of basic operations on arithmetic circuits

In this section, we show that basic operations on arithmetic circuits can be carried out in a uniformity-preserving manner. In particular, we show that polynomial interpolation—a standard tool used to construct low-depth arithmetic circuits—admits a uniform implementation. As a consequence, we obtain uniform families of arithmetic circuits to compute the elementary symmetric polynomials and the inverse of a symbolic Vandermonde matrix.

The main technical tool we use these constructions is an explicit formula for the entries of the inverse of the Vandermonde matrix $\mathrm{Vand}(1, \ldots, n)$ in terms of Stirling numbers of the first kind. Combined with an explicit formula for Stirling numbers, we obtain a circuit family that computes the entries of the inverse of $\mathrm{Vand}(1, \ldots, n)$. We caution the reader that the following proof will be painstakingly detailed, only because it is the first proof that uses that the above notions of uniformity.

Lemma 3.1. *There exists a polylogtime-uniform family of constant-free, constant-depth circuits $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$ over $\mathbb{Q}$ where C_n has no inputs, n^2 outputs, size $\mathrm{poly}(n)$ and computes the entries of the inverse of the Vandermonde matrix $\mathrm{Vand}(1, \ldots, n)$.*

Proof. An explicit formula for the entries of $\mathrm{Vand}^{-1}(1, \ldots, n)$ was given by Macon and Spitzbart [MS58], and the following statement is from [EFP98, Lemma 4]. Letting V_n denote $\mathrm{Vand}(1, \ldots, n)$, we have

$$\left(V_n^{-1}\right)_{j,i} = (-1)^{i+j} \sum_{k=\max(i,j)}^{n} \frac{1}{(k-1)!} \binom{k-1}{i-1} s(k, j), \tag{1}$$

where $s(\cdot, \cdot)$ denotes the Stirling number of the first kind. These Stirling numbers can be computed using the formula

$$s(a, b) = \sum_{j=a}^{2a-b} \binom{j-1}{b-1} \binom{2a-b}{j} \sum_{m=0}^{j-a} \frac{(-1)^{m+a-b} m^{j-b}}{m!(j-a-m)!}, \tag{2}$$

which follows from combining explicit formulas for Stirling numbers of the second kind with symmetric formulas relating the two kinds of Stirling numbers (see [Cha02, Equation 8.21]).

We implement the circuit C_n using these formulas. Our gate names will be tuples of numbers.

1. We start with $2n$ copies of the constants $+1$ and -1. For each $i \in [2n]$, the gates with names $(0, i)$ and $(1, i)$ are constant gates labeled by the constants $+1$ and -1, respectively.

2. Next, we compute $2n$ copies of each number from 1 to $2n$. For each $i, j \in [2n]$, the gate with name $(2, i, j)$ is a $+$ gate whose inputs are the gates with names $(0, k)$ for $k \leq i$. The gate $(2, i, j)$ computes i. The p^{th} predecessor of $(2, i, j)$ is $(0, p)$.

3. We then compute factorials. For each $i \in [2n]$, the gate with name $(3, i)$ is a $\times$ gate whose inputs are the gates named $(1, j)$ for $j \leq i$. The gate $(3, i)$ computes $i!$. The p^{th} predecessor of $(3, i)$ is $(1, p)$.

4. We now compute binomial coefficients. For each $i \in [2n]$ and $j \leq i$, we have gates with names $(4, 1, i, j)$ and $(4, 2, i, j)$. The gate $(4, 1, i, j)$ is a $\times$ gate whose inputs are, in order, the gates $(3, j)$ and $(3, i - j)$. The gate $(4, 2, i, j)$ is a $\div$ gate whose numerator is $(3, i)$ and whose denominator is $(4, 1, i, j)$. Since $(3, i)$ computes $i!$ and $(4, 1, i, j)$ computes $(i - j)! \cdot j!$, the gate $(4, 2, i, j)$ computes $\binom{i}{j}$.

5. Next, we compute numbers of the form i^j as i and j vary over $[2n]$. In particular, for each $i, j \in [2n]$, the gate $(5, i, j)$ is a $\times$ gate with inputs $(2, i, k)$ for each $k \leq j$ (the predecessors will be numbered in this order also). In this way, the gate $(5, i, j)$ computes i^j.

6. Now we compute the Stirling numbers using Eq. (2). For each $a, b \in [n]$ where $b \leq a$, we compute the (j, m) term in the double sum, where $j \in \{a, \ldots, 2a - b\}$ and $m \in \{0, \ldots, j - a\}$, as follows.

   (a) The gate $(6, a, b, j, m, 1)$ is a $\times$ gate that has inputs $(3, m)$ and $(3, j - a - m)$. This gate computes the product of factorials $m! \cdot (j - a - m)!$.

   (b) The gate $(6, a, b, j, m, 2)$ is a $\times$ gate whose first input is $(5, m, j - b)$ and whose second input is either $(0, 1)$ or $(1, 1)$, depending on the parity of $m + a - b$. This gate computes $(-1)^{m+a-b} m^{j-b}$.

   (c) The gate $(6, a, b, j, m, 3)$ is a $\div$ gate with numerator $(6, a, b, j, m, 2)$ and denominator $(6, a, b, j, m, 1)$. This gate computes $\frac{(-1)^{m+a-b} m^{j-b}}{m!(j-a-m)!}$.

   (d) Finally, the gate $(6, a, b, j, m, 4)$ is a $\times$ gate with inputs $(6, a, b, j, m, 3)$, $(4, 2, 2a - b, j)$, and $(4, 2, j - 1, b - 1)$. This gate computes $\binom{j-1}{b-1}\binom{2a-b}{j}\frac{(-1)^{m+a-b}}{m!(j-a-m)!}$, which is precisely the (j, m) term appearing in the double sum used to compute the Stirling number $s(a, b)$.

   For each of the gates above, the predecessors are numbered in the order we have presented them. To compute the Stirling number $s(a, b)$, we add a + gate $(7, a, b)$ whose inputs are $(6, a, b, j, m, 4)$ for all $j \in \{a, \ldots, 2a - b\}$ and $m \in \{0, \ldots, j - a\}$. The predecessors of $(7, a, b)$ are numbered in lexicographic order based on (j, m). Given a, b, and p in binary, computing the p^{th} element in the lexicographic order of the pairs that satisfy the constraint can be done in polynomial time.

7. Finally, we compute the entries of the inverse of the Vandermonde matrix using Eq. (1). For each $i, j \in [n]$, we compute the k^{th} term in the summation, where $k \in \{\max(i, j), \ldots, n\}$, as follows.

   (a) The gate $(8, i, j, k, 1)$ is a $\div$ gate with numerator $(0, 1)$ and denominator $(3, k - 1)$. This gate computes $\frac{1}{(k-1)!}$.

   (b) The gate $(8, i, j, k, 2)$ is a $\times$ gate whose inputs are $(8, i, j, k, 1)$, $(4, 2, k - 1, i - 1)$, and $(7, k, j)$. This gate computes $\frac{1}{(k-1)!}\binom{k-1}{i-1}s(k, j)$.

   To compute the (i, j) entry of V_n^{-1}, we add a + gate $(9, i, j)$ whose inputs are $(8, i, j, k, 2)$ for all $k \in \{\max(i, j), \ldots, n\}$. We add a final $\times$ gate $(10, i, j)$ with inputs $(9, i, j)$ and either $(0, 1)$ or $(1, 1)$, depending on the parity of $i + j$. By Eq. (1), the gate $(10, i, j)$ correctly computes the (j, i) entry of V_n^{-1}.

   Finally, for each of the gates above, its predecessors are numbered in the order we presented them.

From the preceding description of the circuit computing V_n^{-1}, it is clear that the size of the circuit is polynomial in n and the depth is bounded by a universal constant. The above description also serves to bound the uniformity of the circuit family. There are a fixed number of rules that determine the type of a gate and its predecessors. These directly translate into a construction of a Turing machine that can, in polynomial time in $\log(n)$, determine the type of a gate or decide if one gate is a predecessor of another. The only gates with more than a constant number of predecessors are those computing various constants and the gates with names $(7, i, j)$ and $(9, i, j)$. For each of these gates, the circuit description shows how to compute the p^{th} predecessor of a in time polynomial in the binary length of the gate names and the index of the circuit. Therefore we can compute the p^{th} predecessor of a and check if this matches with b to decide the direct connection language. The circuit construction explicitly states the arity of every gate, so this can also be computed efficiently. Finally, the size of the circuit is also apparent from the above construction, therefore the required upper bound on the size can also be computed efficiently. □

Uniformity of interpolation will follow from the above lemma. We start by showing how the coefficients of a single variable can be interpolated, and then handle the general case.

Lemma 3.2. *Let $\mathcal{C} = (C_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of arithmetic circuits over $\mathbb{Q}$ of size $s_{\mathbf{n}}$ and depth bounded by a constant Δ, and with one output. Let $f_{\mathbf{n}}$ be the rational function computed by $C_{\mathbf{n}}$, let y be a distinguished variable, and suppose that every $C_{\mathbf{n}}$ is division-free with respect to y, so that $f_{\mathbf{n}} \in \mathbb{Q}(\mathbf{x})[y]$.*

Let $d_{\mathbf{n}}$ be an upper bound on the degree of $f_{\mathbf{n}}$ in y, and let $f_{\mathbf{n},0}, \ldots, f_{\mathbf{n},d_{\mathbf{n}}} \in \mathbb{Q}(\mathbf{x})$ be such that

$$f_{\mathbf{n}}(\mathbf{x}, y) = \sum_{i=0}^{d_{\mathbf{n}}} f_{\mathbf{n},i}(\mathbf{x}) y^i.$$

Finally, suppose that a binary representation of $d_{\mathbf{n}}$ can be computed from a binary representation of $\mathbf{n}$ in time polynomial in the binary length of $\mathbf{n}$ and $\log(d_{\mathbf{n}})$. Then there exists a polylogtime-uniform family $\mathcal{D} = (D_{\mathbf{n}})_{\mathbf{n}}$ of arithmetic circuits over $\mathbb{Q}$ of size $\text{poly}(s_{\mathbf{n}}, d_{\mathbf{n}})$ and depth at most $\Delta + O(1)$ such that $D_{\mathbf{n}}$ computes $f_{\mathbf{n},0}, \ldots, f_{\mathbf{n},d_{\mathbf{n}}}$.

Proof. Fix $C = C_{\mathbf{n}}$, $f = f_{\mathbf{n}}$, and $d = d_{\mathbf{n}}$. We describe the circuit $D = D_{\mathbf{n}}$.

The circuit D starts by computing the numbers $1, \ldots, d+1$. As in the proof of Lemma 3.1, we have a set of gates labeled by the constant 1, and we use sum gates to compute $1, \ldots, d+1$ from these constants. The constant gates are named $(1, i)$ for $i \in [d+1]$, and the gates computing $1, \ldots, d+1$ are named $(2, i)$ for $i \in [d+1]$. For these sum gates, the p^{th} predecessor of $(2, i)$ is $(1, p)$, as long as $p \leq i$.

The circuit D then contains $d+1$ copies of C. In the i^{th} copy, the input gate y is converted to sum gate with a single input, namely the gate $(2, i)$ that computes the number i. The other input gates of C are also converted to sum gates whose only input is the input gate of D labeled by the same variable. Therefore, these copies of C compute the partial evaluations $f(\mathbf{x}, 1), \ldots, f(\mathbf{x}, d+1)$. The condition that y is not in any denominator of C ensures that the circuit does not divide by zero. The gates in these copies of C are named as follows: if v is the name of a gate in C, then the corresponding gate in the i^{th} copy of C in D is named $(3, i, v)$.

The circuit D also has a copy of the circuit constructed in Lemma 3.1 that computes the inverse of the Vandermonde matrix $\text{Vand}(1, \ldots, d+1)$. We denote this circuit by V. For each gate named v in V, its corresponding copy in D is named $(4, v)$.

The polynomial $f_{\mathbf{n},i}(\mathbf{x})$ is the inner product of a column of the inverse of the Vandermonde matrix, and the outputs of the copies of C. These are computed using $d+1$ product gates and one

sum gate each. For each $i, j \in [d+1]$, the gate $(5, i, j)$ is a $\times$ gate connected to the output of the j^{th} copy of C and the (j, i) entry of the inverse of the Vandermonde matrix, and its predecessors are numbered in this order. For each $i \in [d+1]$, the gate $(6, i)$ is a $+$ gate with inputs $(5, i, j)$ for all $j \in [d+1]$. The p^{th} predecessor of $(6, i)$ is $(5, i, p)$. The gate $(6, i)$ computes the polynomial $f_{\mathbf{n},i}(\mathbf{x})$.

The claims on the size and depth of the circuit family $\mathcal{D}$ follow from the above description and the corresponding bounds on the subcircuits of D. We now argue that $\mathcal{D}$ is a polylogtime-uniform family by describing how the direct connection language is decided.

Let T_C and T_V denote the Turing machines that decide the direct connection languages of C and V, respectively. We will design a Turing machine T_D that decides the direct connection language of D. Let $(\mathbf{n}, a, p, b)$ be an input to T_D. From the first entry in the gate name a, we can determine where in D the gate lies: whether it is part of computing the constants, a gate within a copy of C, part of the circuit that computes the inverse of the Vandermonde matrix, or part of the inner product computation. If a is part of the subcircuit that computes the constants, then deciding if the input is a YES instance is straightforward.

Suppose a is part of the i^{th} copy of C. From a, we can extract the name v_a of the gate within C of which a is a copy. The machine ensures $i \leq d+1$ (recall that $d = d_{\mathbf{n}}$ can be computed in polynomial time in the binary length of $\mathbf{n}$ and $d_{\mathbf{n}}$), and rejects otherwise. If v_a corresponds to the input variable y, then T_D accepts if $p = \epsilon$ and b denotes the type $+$, or if $p = 1$ and b is the gate computing the constant i; all other inputs are rejected. If v_a corresponds to any other input gate, then similarly we accept if either $p = \epsilon$ and b denotes $+$, or if $p = 1$ and b is the input gate in D labeled by the same variable. If v_a is not an input gate and if $p = \epsilon$, then the machine T_D simulates T_C on the input $(\mathbf{n}, v_a, p, b)$ and accepts or rejects accordingly. If $p \neq \epsilon$, then T_D checks if b is also a gate in the i^{th} copy of C. If not, the machine rejects. Otherwise, the machine T_D simulates T_C on $(\mathbf{n}, v_a, p, v_b)$, where v_b is the name of the gate in C corresponding to b.

If a is instead part of the circuit V, we instead will simulate T_V on an appropriate input to decide if $(\mathbf{n}, a, p, b)$ is in the direct connection language of D. This again requires extracting the name of the gate within the circuit V from a, and potentially doing the same for b if $p \neq \epsilon$. The requirement that the binary representation of $d_{\mathbf{n}}$ can be written down in time polynomial in the binary representation of $\mathbf{n}$ ensures that the input to T_V can be computed by T_D in the allotted time.

Finally, if a is part of the inner product subcircuit, then it is either connected to the outputs of the copies of C and V (which is the case for gates labeled $(5, i, j)$) or to other gates in the inner product subcircuit (which is the case for gates labeled $(6, i)$). In the first case, to decide if b is a predecessor of a, we can check if b is an output gate of the correct subcircuit. If $a = (5, i, j)$ then based on whether $p = 1$ or 2, b is either the output of the j^{th} copy of C or the (j, i) entry of the inverse of the Vandermonde matrix. The latter case is handled similarly. The above description explicitly states the arity of every gate, so this can likewise be computed efficiently (with oracle calls to decide the arity of gates in C if required). If $C_{\mathbf{n}}$ has m inputs then $D_{\mathbf{n}}$ has $m - 1$ inputs, and $D_{\mathbf{n}}$ always has $d_{\mathbf{n}} + 1$ outputs. Finally, a bound on the size of $D_{\mathbf{n}}$ follows from a bound on the size of $C_{\mathbf{n}}$, the parameter $d_{\mathbf{n}}$, and the size of the circuit for the inverse of the Vandermonde matrix, therefore such a bound can easily be computed. □

Our proofs of uniformity will often be similar to the one above: we will construct our circuits using previously-constructed uniform subcircuits, along with some other simple machinery. In all cases, the direct connection language will be decided by simulating the Turing machines that decide the direct connection languages of the subcircuits. Therefore, the remaining proofs will not be as low-level as the two proofs of uniformity above.

Uniformity of multivariate interpolation follows from Lemma 3.2 using Kronecker substitution.

Lemma 3.3. *Let $\mathcal{C} = (C_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of arithmetic circuits over $\mathbb{Q}$ of size $s_{\mathbf{n}}$ and depth bounded by a constant Δ, and with one output. Let $f_{\mathbf{n}}$ be the rational function computed by $C_{\mathbf{n}}$, let $\mathbf{y}$ be a distinguished set of $m_{\mathbf{n}}$ variables, and suppose that every $C_{\mathbf{n}}$ is division-free with respect to $\mathbf{y}$, so that $f_{\mathbf{n}} \in \mathbb{Q}(\mathbf{x})[\mathbf{y}]$.*

Let $d_{\mathbf{n}}$ be an upper bound on the degree of $f_{\mathbf{n}}$ in $\mathbf{y}$, and for each $\boldsymbol{\alpha} \in \mathbb{N}^{m_{\mathbf{n}}}$ with $|\boldsymbol{\alpha}| \leq d_{\mathbf{n}}$, let $f_{\mathbf{n},\boldsymbol{\alpha}} \in \mathbb{Q}(\mathbf{x})$ be the coefficient of $\mathbf{y}^{\boldsymbol{\alpha}}$ in $f_{\mathbf{n}}$. Finally, suppose that binary representations of $d_{\mathbf{n}}$ and $m_{\mathbf{n}}$ can be computed from a binary representation of $\mathbf{n}$ in time polynomial in the binary length of $\mathbf{n}$ and $\log(d_{\mathbf{n}})$.

Then there exists a polylogtime-uniform family $\mathcal{D} = (D_{\mathbf{n}})_{\mathbf{n}}$ of arithmetic circuits over $\mathbb{Q}$ of size $\mathrm{poly}(s_{\mathbf{n}}, (d_{\mathbf{n}}+1)^{m_{\mathbf{n}}})$ and depth at most $\Delta + O(1)$ such that $D_{\mathbf{n}}$ computes $f_{\mathbf{n},\boldsymbol{\alpha}}$ for all $\boldsymbol{\alpha} \in \mathbb{N}^{m_{\mathbf{n}}}$ with $|\boldsymbol{\alpha}| \leq d_{\mathbf{n}}$.

Proof. We use Kronecker substitution. Define $g_{\mathbf{n}}$ as

$$g_{\mathbf{n}}(\mathbf{x}, z) \coloneqq f_{\mathbf{n}}\Big(\mathbf{x}, z, z^{d_{\mathbf{n}}+1}, z^{(d_{\mathbf{n}}+1)^2}, \ldots, z^{(d_{\mathbf{n}}+1)^{m_{\mathbf{n}}-1}}\Big).$$

The polynomial $g_{\mathbf{n}}$ has degree at most $d_{\mathbf{n}} \cdot (d_{\mathbf{n}}+1)^{m_{\mathbf{n}}}$ and can be computed by a constant-free circuit of size $\mathrm{poly}(s_{\mathbf{n}}, (d_{\mathbf{n}}+1)^{m_{\mathbf{n}}})$ and depth $\Delta + O(1)$ as follows. We construct a circuit with inputs $\mathbf{x}, z$. Using $+$ gates to copy z and $\times$ gates, we can create gates that compute the powers $z^{(d_{\mathbf{n}}+1)^i}$ for all $i \in [m_{\mathbf{n}} - 1]$. The numbering of the predecessor gates will be the natural one, similar to how we computed powers and factorials in previous proofs.

We then have a copy of the circuit for $f_{\mathbf{n}}$, with all input gates changed to $+$ gates and wired to the inputs $\mathbf{x}$ and these powers of z. The gates are named as in previous proofs: the names indicate what power of z the gate is involved in computing, or that the gate is part of the circuit for $f_{\mathbf{n}}$, in which case the name will contain the name of the corresponding gate within the circuit for $f_{\mathbf{n}}$. This circuit family that computes $\{g_{\mathbf{n}}\}_{\mathbf{n}}$ is polylogtime-uniform. The direct connection language can be decided by simulating the Turing machine for the circuit family $\mathcal{C}$ for gates in the subcircuit that computes $f_{\mathbf{n}}$.

We now apply Lemma 3.2 to this circuit family (the family is division-free with respect to z, and we can compute the binary representation of $d_{\mathbf{n}} \cdot (d_{\mathbf{n}}+1)^{m_{\mathbf{n}}}$ in time polynomial in the binary length of $\mathbf{n}$ and $m_{\mathbf{n}} \log(d_{\mathbf{n}})$). This gives us a circuit family that computes the coefficients of $g_{\mathbf{n}}$ as a polynomial in z, which are in bijection with the coefficients $f_{\mathbf{n},\boldsymbol{\alpha}}$. The gate names in the circuit computing the coefficients of $g_{\mathbf{n}}$ encode the power of z whose coefficient is being computed. Given this, it is easy to compute the monomial $\boldsymbol{\alpha}$ whose coefficient is being computed at each gate: this just involves changing a number to base $d_{\mathbf{n}}+1$ and listing the digits. □

Lemma 3.3 allows us to extract the coefficient of every monomial in the distinguished variables. For some applications, we will need to obtain the coefficients of a given polynomial when treated as a univariate polynomial in many different distinguished variables y. The following lemma shows that this task can be performed uniformly.

Lemma 3.4. *Let $\mathcal{C} = (C_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of arithmetic circuits over $\mathbb{Q}$ of size $s_{\mathbf{n}}$ and depth bounded by a constant Δ. Let $f_{\mathbf{n}}^{(k)}$ be the rational functions computed by $C_{\mathbf{n}}$, for $k \in [m'_{\mathbf{n}}]$, let $\mathbf{y}$ be a distinguished set of $m_{\mathbf{n}}$ variables, and suppose that every $C_{\mathbf{n}}$ is division-free with respect to $\mathbf{y}$, so that each $f_{\mathbf{n}}^{(k)}$ is in $\mathbb{Q}(\mathbf{x})[\mathbf{y}]$.*

Let $d_{\mathbf{n}}$ be an upper bound on the degree of all $f_{\mathbf{n}}^{(k)}$ in $\mathbf{y}$, and for each $0 \leq i \leq d_{\mathbf{n}}$, $j \in [m_{\mathbf{n}}]$ and $k \in [m'_{\mathbf{n}}]$, let $f_{\mathbf{n},j,i}^{(k)} \in \mathbb{Q}(\mathbf{x})[y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_{m_{\mathbf{n}}}]$ be such that

$$f_{\mathbf{n}}^{(k)} = \sum_{i=0}^{d_{\mathbf{n}}} f_{\mathbf{n},j,i}^{(k)} y_j^i.$$

Finally, suppose that binary representations of $d_{\mathbf{n}}$ and $m_{\mathbf{n}}$ can be computed from a binary representation of $\mathbf{n}$ in time polynomial in the binary length of $\mathbf{n}$ and $\log(d_{\mathbf{n}})$.

Then there exists a polylogtime-uniform family $\mathcal{D} = (D_{\mathbf{n}})_{\mathbf{n}}$ of arithmetic circuits over $\mathbb{Q}$ of size $\mathrm{poly}(s_{\mathbf{n}}, m_{\mathbf{n}}, d_{\mathbf{n}})$ and depth $\Delta + O(1)$ such that $D_{\mathbf{n}}$ computes $f^{(k)}_{\mathbf{n},j,i}$ for all $0 \leq i \leq d_{\mathbf{n}}$, $j \in [m_{\mathbf{n}}]$ and $k \in [m'_{\mathbf{n}}]$.

Proof. This proof is similar to that of Lemma 3.2. Fix $C = C_{\mathbf{n}}$, $f = f_{\mathbf{n}}$, and $d = d_{\mathbf{n}}$. We describe the circuit $D = D_{\mathbf{n}}$.

The circuit D has set of gates named $(1, i)$ that are constant gates labeled by the constant 1, and a set of $+$ gates $(2, i)$ for $i \in [d]$ that compute the numbers $1, \ldots, d+1$. The predecessors are numbered the obvious way. The circuit D then has $m_{\mathbf{n}}(d_{\mathbf{n}} + 1)$ copies of C, one for every pair (i, j) with $0 \leq i \leq d_{\mathbf{n}}$ and $1 \leq j \leq m_{\mathbf{n}}$. The gates in the $(i, j)^{\text{th}}$ copy are named $(3, i, j, v)$, where v is the name of the corresponding gate in C. In the $(i, j)^{\text{th}}$ copy of C, the input gates labeled by y_j are replaced by an arity-one summation gate whose child is the gate $(2, i)$ computing the integer i.

The circuit D then has a copy of the circuit V from Lemma 3.1 that computes the inverse of the Vandermonde matrix $\mathrm{Vand}(1, \ldots, d_{\mathbf{n}} + 1)$. These gates are named $(4, v)$, where v is the name of the corresponding gate in V. Following this, the circuit D has gates $(5, k, *)$ and $(6, k, *)$ that compute all required products of the evaluations and the matrix inverse. Again the predecessors will be numbered just as in the proof of Lemma 3.2.

The claimed bounds on the size and depth of D follow from the description above. It remains to show how the direct connection language of D is decided. Let $(\mathbf{n}, a, p, b)$ be an input to the machine deciding the direct connection language. If a is of the form $(t, *)$ with $t \neq 3$, then deciding if this is a YES instance can be done the exact same way as in the proof of Lemma 3.2. If a is of the form $(3, i, j, v)$, and if v is an input gate within C, we can use the indices i, j to ensure that the input is wired to the correct constant or variable (we first check that $0 \leq i \leq d_{\mathbf{n}}$ and $1 \leq j \leq m_{\mathbf{n}}$, which takes time polynomial in the binary length of $\mathbf{n}$, by assumption). In particular, if the input is x_t with $t \neq j$, then the gate must be wired to x_t itself, otherwise if $t = j$ then the input is wired to the gate computing the constant i. The rest of the computation is also the exact same as in the proof of Lemma 3.2. □

A corollary of the above interpolation results is a uniform family of circuits that compute the elementary symmetric polynomials.

Lemma 3.5. *There exists a polylogtime-uniform family of constant-free, weakly division-free, constant-depth arithmetic circuits $\mathcal{C} = (C_n)_n$ over $\mathbb{Q}$ such that C_n has size $\mathrm{poly}(n)$ and computes the elementary symmetric polynomials $e_1, \ldots, e_n$ on n variables.*

Proof. We use the constant-depth circuits for the elementary symmetric polynomials designed by Ben-Or. There exists a polylogtime-uniform family $\mathcal{D} = (D_n)_n$ of constant-free, polynomial-size, constant-depth circuits over $\mathbb{Q}$ such that D_n has $n+1$ inputs $x_1, \ldots, x_n, t$, and computes $\prod_{i=1}^{n}(x_i + t)$. Indeed D_n can just compute this polynomial using n sum gates and a single product gate. The degree of D_n is exactly n, and $\mathcal{D}$ is division-free. Therefore, $\mathcal{D}$ satisfies all the assumptions in Lemma 3.2, and we can invoke Lemma 3.2 to obtain $\mathcal{C}$. □

Using Lemma 3.5, we can also construct a uniform family of circuits that compute the inverse of a generic Vandermonde matrix.

Lemma 3.6. *There exists a polylogtime-uniform family of constant-free, constant-depth arithmetic circuits $\mathcal{C} = (C_n)_n$ over $\mathbb{Q}$ such that C_n has size $\mathrm{poly}(n)$ and computes the entries of the inverse of the matrix $\mathrm{Vand}(x_1, \ldots, x_n)$.*

Proof. The circuit uses the following two facts: the determinant of $V := \mathrm{Vand}(x_1, \ldots, x_n)$ is $\prod_{i<j}(x_i - x_j)$, and the entries of the inverse of this matrix can be written as

$$V^{-1}_{i,j} = \frac{(-1)^{n-i} e_{n-i}(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)}{\prod_{k \neq j}(x_j - x_k)}.$$

We now describe the circuit C_n. This circuit contains n copies of the circuit for the elementary symmetric polynomials in $n-1$ variables constructed in Lemma 3.5. To the j^{th} copy, the inputs provided are $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$. The circuit also has n subcircuits where the j^{th} one computes $\prod_{k \neq j}(x_j - x_k)$. Finally, the circuit combines these to compute the entries of the inverse. The claims on the size and depth are straightforward given the above description. For uniformity, the argument is essentially the same as the one in the proof of Lemma 3.2: the Turing machine that has to decide the direct connection language of $\mathcal{C}$ can simulate the Turing machines that decide the direct connection languages of the constituent subcircuits. Note that the output gate that computes the entry at position (i, j) in the inverse will have (i, j) encoded in its name. □

4 The multivariate resultant

This section discusses the theory of resultants. We first introduce the multivariate resultant, which is a generalization of the usual resultant to multiple polynomials in several variables that tests when a system of homogeneous polynomial equations has a solution. We will then discuss how the multivariate resultant can be used to test when systems of inhomogeneous equations have solutions. Finally, we will discuss how the multivariate resultant can be used to count the number of solutions in zero-dimensional systems.

For a friendly introduction to the multivariate resultant, we refer the reader to Cox, Little, and O'Shea [CLO05, Chapter 3]. A comprehensive treatment of resultants can be found in the book of Gelfand, Kapranov, and Zelevinsky [GKZ94], in the survey of Ierardi and Kozen [IK93], or in the work of Jouanolou [Jou91].

4.1 Multivariate resultants

The resultant of two univariate polynomials f, g is a polynomial in the coefficients of f and g that vanishes exactly when f and g have a common solution in the algebraic closure $\overline{\mathbb{F}}$ of the base field $\mathbb{F}$. The multivariate resultant generalizes this, and detects when a homogeneous system of equations has projective solutions.

Throughout this section, we fix a natural number $n \in \mathbb{N}$ and a choice of natural numbers $d_0, \ldots, d_n \in \mathbb{N}$. For $d \in \mathbb{N}$, we write $M_{n,d}$ for the set of homogeneous degree-d monomials in the $n+1$ variables $x_0, \ldots, x_n$. We take $\mathbf{u} = \{u_{i,\boldsymbol{\alpha}} \mid i \in \{0, \ldots, n\}, \boldsymbol{\alpha} \in M_{n,d}\}$ to be a set of variables corresponding to the coefficients of $n+1$ homogeneous polynomials of degrees $d_0, \ldots, d_n$, respectively.

With this notation in hand, we now define the multivariate resultant.

Definition 4.1 (see, e.g., [Jou91, Proposition 2.3])**.** Let $\mathbb{F}$ be any field. The *resultant* $\mathrm{Res}_{d_0,\ldots,d_n} \in \mathbb{F}[\mathbf{u}]$ is the unique polynomial satisfying the following conditions.

1. If $F_0, \ldots, F_n \in \mathbb{F}[x_0, \ldots, x_n]$ are homogeneous polynomials of degrees $d_0, \ldots, d_n$, respectively, then $\mathrm{Res}_{d_0,\ldots,d_n}(F_0, \ldots, F_n) = 0$ if and only if the system of equations $F_0(\mathbf{x}) = \cdots = F_n(\mathbf{x}) = 0$ has a solution in $\mathbb{P}^n_{\overline{\mathbb{F}}}$. By $\mathrm{Res}_{d_0,\ldots,d_n}(F_0, \ldots, F_n)$ we mean the evaluation of $\mathrm{Res}_{d_0,\ldots,d_n}$ obtained by setting $u_{i,\boldsymbol{\alpha}}$ equal to the coefficient of the monomial $\mathbf{x}^{\boldsymbol{\alpha}}$ in F_i.

2. $\mathrm{Res}_{d_0,\dots,d_n}(x_0^{d_0},\dots,x_n^{d_n}) = 1$.

3. The polynomial $\mathrm{Res}_{d_0,\dots,d_n}$ is irreducible in $\overline{\mathbb{F}}[\mathbf{u}]$.

Moreover, if we write $\mathrm{Res}^{\mathbb{Q}}_{d_0,\dots,d_n}$ and $\mathrm{Res}^{\mathbb{F}}_{d_0,\dots,d_n}$ for the resultants over $\mathbb{Q}$ and $\mathbb{F}$, respectively, then the former has integer coefficients and $\mathrm{Res}^{\mathbb{F}}_{d_0,\dots,d_n}$ is the image of $\mathrm{Res}^{\mathbb{Q}}_{d_0,\dots,d_n}$ under the natural ring homomorphism $\mathbb{Z}[\mathbf{u}] \to \mathbb{F}[\mathbf{u}]$. ◊

When n and the degrees $d_0,\dots,d_n$ are clear from context, we will drop the subscript and use Res to denote the resultant $\mathrm{Res}_{d_0,\dots,d_n}$. The existence of a polynomial satisfying Definition 4.1 is not obvious. We refer the reader interested in the existence of the resultant to Jouanolou [Jou91]. The final statement in Definition 4.1, the fact that the resultant over any field is the image of the resultant over the integers, will be crucial in our proofs.

The multivariate resultant generalizes the determinant. Consider a collection of homogeneous linear forms $L_0,\dots,L_n \in \mathbb{F}[x_0,\dots,x_n]$ given by $L_i(\mathbf{x}) = \sum_{j=0}^n a_{i,j}x_j$. The system of equations $L_0(\mathbf{x}) = \cdots = L_n(\mathbf{x}) = 0$ has a nonzero solution exactly when

$$\det\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,1} & \cdots & a_{n,n} \end{pmatrix} = 0.$$

Likewise, from Definition 4.1, we know that this system has a nonzero solution exactly when $\mathrm{Res}_{1,\dots,1}(L_0,\dots,L_n) = 0$. This is no coincidence: the polynomials $\mathrm{Res}_{1,\dots,1}(\mathbf{u})$ and $\det_{n+1}(\mathbf{u})$ are the same!

It is clear from the definition of the resultant that it is a polynomial in $\sum_{i=0}^n \binom{n+d_i}{d_i}$ variables. Less obvious is its degree, which is provided by the following lemma.

Lemma 4.2 (see, e.g., [CLO05, Chapter 3, Theorem 3.1]). *The resultant* $\mathrm{Res}_{d_0,\dots,d_n}(\mathbf{u})$ *is homogeneous of degree* $d_0\cdots d_{i-1}d_{i+1}\cdots d_n$ *with respect to the variables* $\{u_{i,\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in M_{n,d_i}\}$ *and is homogeneous of total degree* $\sum_{i=0}^n d_0\cdots d_{i-1}d_{i+1}\cdots d_n$.

Our work will focus on the design of uniform circuits of constant depth and size $d^{\mathrm{poly}(n)}$ that compute the resultant. To design such circuits, we will use non-determinantal formulations of the resultant, since determinants provably require super-polynomial size to compute using circuits of bounded depth [LST21; For24] and are conjectured to not be computable by quasipolynomial-size circuits of bounded depth. In particular, we will make use of the following identity, known as the *Poisson formula*, that expresses the resultant $\mathrm{Res}_{d_0,\dots,d_n}$ as a product of two terms: one is the smaller resultant $\mathrm{Res}_{d_0,\dots,d_{n-1}}$, and the other is related to the values attained by F_n on the common zeroes of $F_0,\dots,F_{n-1}$. The special case of the Poisson formula corresponding to $n = 1$ was used to design constant-depth circuits for the bivariate resultant [AW24; BKR+25b]; its multivariate generalization was already used by Jeronimo and Sabia [JS07], and will likewise be key in designing low-depth circuits for the multivariate resultant.

Theorem 4.3 (see, e.g., [CLO05, Chapter 3, Section 3, Exercise 8]). *Let* $F_0,\dots,F_n \in \mathbb{F}[\mathbf{x}]$ *be homogeneous polynomials of degrees* $d_0,\dots,d_n$, *respectively. For* $i \in \{0,1,\dots,n\}$, *let*

$$\overline{F}_i(\mathbf{x}) \coloneqq F_i(x_0,\dots,x_{n-1},0)$$
$$f_i(\mathbf{x}) \coloneqq F_i(x_0,\dots,x_{n-1},1).$$

Suppose $\mathrm{Res}_{d_0,\ldots,d_{n-1}}(\overline{F}_0,\ldots,\overline{F}_{n-1}) \neq 0$. *Then the resultant* $\mathrm{Res}_{d_0,\ldots,d_n}(F_0,\ldots,F_n)$ *satisfies the identity*

$$\mathrm{Res}_{d_0,\ldots,d_n}(F_0,\ldots,F_n) = \mathrm{Res}_{d_0,\ldots,d_{n-1}}(\overline{F}_0,\ldots,\overline{F}_{n-1})^{d_n} \cdot \prod_{\boldsymbol{\alpha}\in \mathrm{V}(f_0,\ldots,f_{n-1})} f_n(\boldsymbol{\alpha})^{m(\boldsymbol{\alpha})},$$

where $\mathrm{V}(f_0,\ldots,f_{n-1}) \subseteq \overline{\mathbb{F}}^n$ *is the finite set of common zeroes of* $f_0,\ldots,f_{n-1}$ *in the algebraic closure* $\overline{\mathbb{F}}^n$ *and* $m(\boldsymbol{\alpha}) \in \mathbb{N}$ *is the multiplicity of* $\boldsymbol{\alpha}$ *in* $\mathrm{V}(f_0,\ldots,f_{n-1})$.

4.2 Satisfiability of affine systems

The resultant naturally suggests an algorithm to decide the satisfiability (in the algebraic closure of the coefficient field) of homogeneous systems of $n+1$ equations in $n+1$ variables, which are known as *square* systems. Given equations $F_0(\mathbf{x}) = F_1(\mathbf{x}) = \cdots = F_n(\mathbf{x}) = 0$, we simply compute the resultant $\mathrm{Res}(F_0,\ldots,F_n)$ and report that this system is satisfiable if and only if $\mathrm{Res}(F_0,\ldots,F_n) = 0$. To solve a larger system $F_0 = \cdots = F_m = 0$ of homogeneous equations, one can reduce to the square case by forming $n+1$ random linear combinations of the F_i. With good probability, the resulting square system will be equisatisfiable with the original system, so algorithms that compute the resultant allow us to decide the satisfiability of homogeneous systems of equations of any size. What about systems of inhomogeneous equations?

A natural strategy to solve a system of inhomogeneous equations is to homogenize the system and attempt to use the resultant. This strategy works for some, but not all, affine systems, depending on the behaviour at infinity of the homogenized equations. A classical method for using the multivariate resultant to study affine systems is the method of generalized characteristic polynomials, introduced by Canny [Can90]. Here, instead of just homogenizing an affine system, an additional perturbation is used to eliminate degenerate behaviour at infinity. Using this method, Ierardi [Ier89a] obtained a clean reduction from the task of testing satisfiability of affine systems to computing multivariate resultants of polynomials with coefficients in small polynomial rings. In this subsection, we recall Ierardi's reduction. We direct the reader to Ierardi's PhD thesis ([Ier89b]) for an explanation of the rather beautiful geometric ideas that underlie the reduction and related constructions.

We now quote Ierardi's reduction from testing satisfiability of affine systems to computing multivariate resultants. In the following result and in the rest of this section, we make the following assumptions:

(A) the field $\mathbb{F}$ is one of $\mathbb{Q}$, $\mathbb{F}_{p^a}$, $\mathbb{Q}(y_1,\ldots,y_k)$ or $\mathbb{F}_{p^a}(y_1,\ldots,y_k)$

(B) in the latter two cases, the input to our algorithms have coefficients that are polynomial in $\mathbf{y}$.

In particular, the height of such input, as introduced in Section 2.3, is always well-defined.

Proposition 4.4 ([Ier89a]). *Suppose that (A) and (B) hold, and let* $f_1,\ldots,f_m \in \mathbb{F}[x_1,\ldots,x_n]$ *be polynomials of degree at most* d *and height at most* h. *Suppose the coefficient subfield of* $\mathbb{F}$ *has at least* $15nd^n$ *elements. There is a polynomial-time Monte Carlo algorithm with success probability* $2/3$ *that takes as inputs* $f_1,\ldots,f_m$, *and produces a set of polynomials* $G_{i,j} \in \mathbb{F}[t,w,u,x_0,\ldots,x_n]$ *with* $0 \leq i \leq n$ *and* $1 \leq j \leq n$ *with the following properties.*

- *Each* $G_{i,j}$ *is homogeneous in* $x_0,\ldots,x_n$ *of degree at most* d.
- *Each* $G_{i,j}$ *has degree at most* n *in* w *and degree at most one in* t *and* u.
- *Each* $G_{i,j}$ *has height at most* $h \cdot (n \log d)^c$ *for a universal constant* c.

- *The variety* $\mathrm{V}(f_1, \ldots, f_m)$ *is nonempty if and only if there exists a* j *such that*

$$(\mathrm{TT}_w \, \mathrm{TT}_t \, \mathrm{Res}(G_{0,j}, \ldots, G_{n,j}))(0) = 0,$$

  where the resultant $\mathrm{Res}(G_{0,j}, \ldots, G_{n,j})$ *is computed by regarding the* $G_{i,j}$ *as polynomials in* $\mathbf{x}$ *with coefficients in* $\mathbb{F}(t, w, u)$.

The proof of the above reduction is essentially the content of [Ier89a, §§2–3]. We provide an outline of the proof below, although we import a technical statement regarding the multivariate resultant without proof (Lemma 4.7). This technical lemma will also be useful when we show how the resultant can be used to count solutions in a zero dimensional system.

We start by defining a notion of limit set for projective varieties.

Definition 4.5. Let $G_1, \ldots, G_n \in \mathbb{F}[x_0, \ldots, x_n]$ be forms of degrees $d_1, \ldots, d_n$ respectively. Let $V_0 \coloneqq \mathrm{V}(G_1, \ldots, G_n)$ be the variety defined by $G_1, \ldots, G_n$. The *limit set of* V_0 is the subvariety V_0^* of V_0 obtained by the following procedure.

Let t be a new variable. Define $\widehat{G_i} \coloneqq G_i + t \cdot x_i^{d_i}$ and let $V \coloneqq \mathrm{V}\left(\widehat{G_1}, \ldots, \widehat{G_n}\right) \subseteq \mathbb{P}^n \times \mathbb{A}$. Let V^* denote the union of the components of V that are not contained within a subspace of the form $\mathrm{V}(t - \tau)$ for any $\tau \in \mathbb{F}$. Equivalently, these are the components of V whose projection onto $\mathbb{A}$ is surjective. Define $V_0^* \coloneqq V^* \cap \mathrm{V}(t = 0)$, which we treat as a variety in $\mathbb{P}^n$. ◊

Lemma 4.6. *For any forms* $G_1, \ldots, G_n \in \mathbb{F}[x_0, \ldots, x_n]$ *of degrees* $d_1, \ldots, d_n$*, the limit set* V_0^* *is a finite subset of* $\mathrm{V}(G_1, \ldots, G_n)$ *that contains all its isolated points. The set* V_0^* *has size at most* $\prod_{i=1}^n d_i$.

Proof. The first statement is the content of [Ier89a, Lemma 2.3]. The second statement follows from Bézout's inequality ([BCS97, Theorem 8.28]) applied to $\widehat{G_1}, \ldots, \widehat{G_n}$: since V_0^* is finite, its cardinality is bounded above by the Bézout number of $\widehat{G_1}, \ldots, \widehat{G_n}$ seen as polynomials in $\mathbb{F}(t)[x_0, \ldots, x_n]$. □

We now quote the technical lemma from Ierardi [Ier89a] that produces a polynomial R_0^* whose factors are linear polynomials that correspond to the points of the limit set V_0^*.

Lemma 4.7 ([Ier89a, Lemma 2.6])**.** *Let* $G_1, \ldots, G_n \in \mathbb{F}[x_0, \ldots, x_n]$ *be forms of degrees* $d_1, \ldots, d_n$*. Let* $F_1, \ldots, F_m \in \mathbb{F}[x_0, \ldots, x_n]$ *be additional homogeneous forms of degree* d*. Define* $\widehat{G_i} \coloneqq G_i + t x_i^{d_i}$*. Define* $L \in \mathbb{F}[x_0, \ldots, x_n, u_0, \ldots, u_n, v_1, \ldots, v_m, t]$ *as* $L \coloneqq \sum_{i=0}^n u_i x_i^d + \sum_{j=1}^m v_j F_j$*. Define*

$$R_0^* \coloneqq \mathrm{TT}_t \, \mathrm{Res}\left(\widehat{G_1}, \ldots, \widehat{G_n}, L\right),$$

where the resultant $\mathrm{Res}\left(\widehat{G_1}, \ldots, \widehat{G_n}, L\right)$ *is computed by viewing the* G_i *and* L *as polynomials in* $\mathbf{x}$ *with coefficients in* $\mathbb{F}(\mathbf{u}, \mathbf{v}, t)$.

Then the polynomial $R_0^* \in \mathbb{F}[\mathbf{u}, \mathbf{v}]$ *factors into a product of linear forms over* $\overline{\mathbb{F}}$*. For each point* $\boldsymbol{\alpha} \in V_0^*$ *in the limit set of* $\mathrm{V}(G_1, \ldots, G_n)$*, the linear polynomial* $\sum_{i=0}^n u_i \alpha_i^d + \sum_{j=1}^m v_j F_j(\boldsymbol{\alpha})$ *is a factor of* R_0^**. Moreover, every factor of* R_0^* *has this structure for some* $\boldsymbol{\alpha} \in V_0^*$.

The above lemma generalizes the method of U-resultants for zero-dimensional projective systems. As a consequence of Lemma 4.7, we can use resultants to detect isolated points of an affine system of equations.

Lemma 4.8. *Let $B \subseteq \mathbb{F} \setminus \{0\}$ be a subset of the coefficient subfield of $\mathbb{F}$. Let $g_1, \dots, g_n, f_1, \dots, f_m \in \mathbb{F}[x_1, \dots, x_n]$ be polynomials of degree at most d. Let $\gamma_1, \dots, \gamma_m$ be nonzero field elements picked independently and uniformly at random from B. Define F_i and G_i to be the homogenizations of the polynomials f_i and g_i, respectively, with respect to a new variable x_0. Define $\widehat{G_i} \coloneqq G_i + t x_i^{\deg g_i}$ and $H \coloneqq u_0 x_0^d + \sum_{i=1}^n w^i x_i^d + \sum_{j=1}^m \gamma_j x_0^{d - \deg F_j} F_j$. Define*

$$\beta \coloneqq \Big(\mathrm{TT}_w \, \mathrm{TT}_t \operatorname{Res}\Big(\widehat{G_1}, \dots, \widehat{G_n}, H\Big)\Big)(0),$$

where the resultant $\operatorname{Res}\Big(\widehat{G_1}, \dots, \widehat{G_n}, H\Big)$ is computed by viewing the $\widehat{G_i}$ and H as polynomials in $\mathbf{x}$ with coefficients in $\mathbb{F}(u_0, w, t)$.

With probability at least $1 - d^n/|B|$ over the choice of γ, the following holds. If $\beta = 0$, then $\mathrm{V}(f_1, \dots, f_m, g_1, \dots, g_n) \neq \varnothing$, and in particular $\mathrm{V}(f_1, \dots, f_m) \neq \varnothing$. Conversely, if there exists an isolated point $\boldsymbol{\alpha}' \in \mathrm{V}(g_1, \dots, g_n)$ such that $\boldsymbol{\alpha}' \in \mathrm{V}(f_1, \dots, f_m)$, then $\beta = 0$.

Proof. Define $L \coloneqq u_0 x_0^d + \sum_{i=1}^n u_i x_i^d + \sum_{j=1}^m v_j x_0^{d - \deg F_j} F_j$, where the u_i and v_j are variables. Let

$$R \coloneqq \operatorname{Res}\Big(\widehat{G_1}, \dots, \widehat{G_n}, L\Big),$$

where this resultant is computed by viewing the $\widehat{G_i}$ and L as polynomials in $\mathbf{x}$ with coefficients in $\mathbb{F}(\mathbf{u}, \mathbf{v}, t)$. Let $R_0^* \coloneqq \mathrm{TT}_t \, R$ and let V_0^* be the limit set (as defined in Definition 4.5) of $G_1, \dots, G_n$.

By Lemma 4.7, we know that R_0^* factors into linear forms that look like

$$\sum_{i=0}^n u_i \alpha_i^d + \sum_{j=1}^m v_j \alpha_0^{d - \deg F_j} F_j(\boldsymbol{\alpha}),$$

where $\boldsymbol{\alpha} \in V_0^*$ is a point in the limit set of $G_1, \dots, G_n$. If there is a factor of R_0^* that is independent of $v_1, \dots, v_m$, then the corresponding point $\boldsymbol{\alpha}$ lies in $\mathrm{V}\Big(x_0^{d - \deg F_1} F_1, \dots, x_0^{d - \deg F_m} F_m\Big)$. If this factor further depends on u_0, then we can write $\boldsymbol{\alpha} = (1, \boldsymbol{\alpha}')$, and the point $\boldsymbol{\alpha}'$ lies in $\mathrm{V}(g_1, \dots, g_n, f_1, \dots, f_m)$. Conversely, if there is an isolated point $\boldsymbol{\alpha}' \in \mathrm{V}(g_1, \dots, g_n)$ such that $\boldsymbol{\alpha}' \in \mathrm{V}(f_1, \dots, f_m)$, then $(1, \boldsymbol{\alpha}') \in V_0^*$ and the corresponding factor of R_0^* will depend on u_0 and be independent of $v_1, \dots, v_m$.

Now let r denote the resultant

$$\operatorname{Res}\Big(\widehat{G_1}, \dots, \widehat{G_n}, H\Big),$$

where the polynomials $\widehat{G_i}$ and H are viewed as polynomials in $\mathbf{x}$ with coefficients in $\mathbb{F}(u_0, w, t)$. The form H is obtained from L by substituting $u_i \mapsto w^i$ for $i \in [n]$ and $v_j \mapsto \gamma_j$ for $j \in [m]$. Since the resultant is a polynomial function of the coefficients, we have $r = R(t, u_0, w, w^2, \dots, w^n, \gamma_1, \dots, \gamma_m)$. Under the same substitution, a factor of R_0^* corresponding to $\boldsymbol{\alpha} \in V_0^*$ is mapped to

$$u_0 \alpha_0^d + \sum_{i=1}^n w^i \alpha_i^d + \sum_{j=1}^m \gamma_j \alpha_0^{d - \deg F_j} F_j(\boldsymbol{\alpha}).$$

Each factor of R_0^* depends on at least one of $u_0, \dots, u_n$, since $\boldsymbol{\alpha}$ is a point in projective space. Combined with the fact that the polynomials $u_0, w, \dots, w^n$ are linearly independent, it follows that no factor of R_0^* is mapped to zero under the above substitution. In particular, R_0^* is nonzero under this substitution. This implies that $\mathrm{TT}_t \, r = R_0^*(u_0, w, w^2, \dots, w^n, \gamma_1, \dots, \gamma_n)$. Define $r_0^* \coloneqq \mathrm{TT}_t \, r$. We have $\beta = (\mathrm{TT}_w \, r_0^*)(0)$ by definition.

Any factor of R_0^* that depends on one of the variables in $\mathbf{v}$ is mapped to a polynomial with nonzero constant term with probability at least $1 - 1/|B|$. Since V_0^* has size at most d^n (Lemma 4.6), every such factor of R_0^* is mapped to a polynomial with a nonzero constant term with probability at least $1 - d^n/|B|$. We show that the conclusion of the lemma holds whenever this event occurs.

Assume therefore that the event occurs. Suppose $\beta = 0$. This implies that some factor of r_0^* not divisible by w has a constant term of zero. By assumption, every factor of R_0^* that depends on a variable in $\mathbf{v}$ results in a factor of r_0^* that has a nonzero constant term, so it must be the case that some factor $\hat{R}$ of R_0^* does not depend on any variable in $\mathbf{v}$. Moreover, the factor $\hat{R}$ depends on u_0, since the corresponding factor of r_0^* is not divisible by w. Together, these conditions imply that the point $\boldsymbol{\alpha} \in V_0^* \subseteq V(G_1, \ldots, G_n)$ corresponding to $\hat{R}$ satisfies $\alpha_0 \neq 0$ and $F_j(\boldsymbol{\alpha}) = 0$ for all $j \in [m]$. This means that $\boldsymbol{\alpha}' \in \mathrm{V}(g_1, \ldots, g_n, f_1, \ldots, f_m)$, where $\boldsymbol{\alpha} = (1, \boldsymbol{\alpha}')$. This set is thus nonempty as claimed.

Conversely, suppose there is an isolated point $\boldsymbol{\alpha}' \in \mathrm{V}(g_1, \ldots, g_n)$ such that $\boldsymbol{\alpha}' \in \mathrm{V}(f_1, \ldots, f_m)$. Because $\boldsymbol{\alpha}'$ is an isolated point of $\mathrm{V}(g_1, \ldots, g_n)$, we have $(1, \boldsymbol{\alpha}') \in V_0^*$, so there is a factor of R_0^*, and thus a factor of r_0^*, corresponding to $(1, \boldsymbol{\alpha}')$. The corresponding factor of r_0^* has the form

$$u_0 + \sum_{i=1}^{n} w^i \alpha_i^d.$$

This polynomial is not divisible by w and has a constant term of zero, so it follows that $\beta = 0$, as desired. □

With Lemma 4.8, we can finish the proof of Proposition 4.4. The argument will require the following standard results regarding random hyperplane sections and varieties defined by random linear combinations of a given set of polynomials.

Lemma 4.9. *Let $B \subseteq \mathbb{F}$ be a finite set. Let $V \subseteq \mathbb{P}^n$ be a projective variety of dimension r and degree D. Suppose ℓ is a linear form with each coefficient picked independently and uniformly at random from B. Then with probability at least $1 - D/|B|$, the intersection $V \cap \mathrm{V}(\ell)$ has dimension $r - 1$. If $\ell_1, \ldots, \ell_{r+1}$ are linear forms chosen the same way, then $V \cap \mathrm{V}(\ell_1, \ldots, \ell_{r+1}) = \varnothing$ with probability at least $1 - (r+1)D/|B|$.*

Let $W \subseteq \mathbb{A}^n$ be a projective variety of dimension r and degree D. Suppose ℓ is a linear polynomial with each coefficient picked independently and uniformly at random from B. Then with probability at least $1 - 2D/|B|$, the intersection $W \cap \mathrm{V}(\ell)$ has dimension $r - 1$. If $\ell_1, \ldots, \ell_{r+1}$ are linear polynomials chosen the same way, then $W \cap \mathrm{V}(\ell_1, \ldots, \ell_{r+1}) = \varnothing$ with probability at least $1 - 2(r+1)D/|B|$.

Lemma 4.10. *Let $B \subseteq \mathbb{F}$ be a finite set. Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be polynomials of degree at most d. Suppose $g_1, \ldots, g_{n+1}$ are linear combinations of $f_1, \ldots, f_m$, with each coefficient picked independently and uniformly at random from B. Then with probability at least $1 - (n+1)d^n/|B|$, the following is true. For each $s \in [n+1]$, every irreducible component of $\mathrm{V}(g_1, \ldots, g_s)$ of codimension less than s is a component of $\mathrm{V}(f_1, \ldots, f_m)$. As a consequence, $\mathrm{V}(g_1, \ldots, g_{n+1}) = \mathrm{V}(f_1, \ldots, f_m)$. The analogous result in the projective setting also holds.*

The proof of both Lemma 4.9 and Lemma 4.10 are implicit in the methods of Heintz [Hei83]. For an explicit proof of the latter, see for example [KP96, Lemma 36]. The former can be proved using essentially the same arguments.

Proof of Proposition 4.4. If the coefficient subfield of $\mathbb{F}$ is $\mathbb{Q}$, let B be the set of natural numbers $\{1, \ldots, 15nd^n\}$. If the coefficient subfield of $\mathbb{F}$ is finite, let B be the set of nonzero elements of the coefficient subfield. For a fixed $j \in [n]$, we construct $G_{0,j}, \ldots, G_{n,j}$ as follows. Let $g_{1,j}, \ldots, g_{j,j}$

be random linear combinations of $f_1, \dots, f_m$, where the coefficients are picked independently and uniformly from B. Let $g_{j+1,j}, \dots, g_{n,j}$ be random linear polynomials, with coefficients picked independently and uniformly from B. For $i \in [n]$, let $G_{i,j}$ be defined as $G_{i,j} \coloneqq g_{i,j}^h + t x_i^{\deg g_{i,j}}$, where $g_{i,j}^h$ is the homogenization of $g_{i,j}$ with respect to a new variable x_0. To construct $G_{0,j}$, we pick elements $\gamma_{1,j}, \dots, \gamma_{m,j}$ independently and uniformly at random from B, and set $G_{0,j} \coloneqq u_0 x_0^d + \sum_{i=1}^n w^i x_i^d + \sum_{k=1}^m \gamma_{k,j} x_0^{d-\deg f_k} F_k$, where F_k is the homogenization of f_k with respect to the variable x_0. These polynomials have the claimed degree and height bounds. For each j, define

$$\beta_j \coloneqq (\mathrm{TT}_w \, \mathrm{TT}_t \, \mathrm{Res}(G_{1,j}, \dots, G_{n,j}, G_{0,j}))(0).$$

With probability at least $1 - nd^n/|B|$, the conclusion of Lemma 4.8 holds for all the β_j. For the rest of the proof, we assume that this event occurs. If $\beta_j = 0$ for any j, then $\mathrm{V}(f_1, \dots, f_m) \neq \varnothing$. To complete the proof, it suffices to show that if $\mathrm{V}(f_1, \dots, f_m) \neq \varnothing$, then $\beta_j = 0$ for some j. We show that this happens with high probability.

To this end, let $s \coloneqq n - \dim \mathrm{V}(f_1, \dots, f_m)$. By Lemma 4.10, with probability at least $1 - 2nd^n/|B|$, the polynomials $g_{1,s}, \dots, g_{s,s}$ define an equidimensional variety of codimension s. Each component of $\mathrm{V}(f_1, \dots, f_m)$ is contained in some component of $\mathrm{V}(g_{1,s}, \dots, g_{s,s})$. In particular, each component of $\mathrm{V}(f_1, \dots, f_m)$ of maximal dimension is a component of $\mathrm{V}(g_{1,s}, \dots, g_{s,s})$.

Since $g_{s+1,s}, \dots, g_{n,s}$ are linear polynomials with coefficients from B, by Lemma 4.9 it holds that

$$\dim(\mathrm{V}(g_{1,s}, \dots, g_{s,s}) \cap \mathrm{V}(g_{s+1,s}, \dots, g_{n,s})) = 0$$

with probability at least $1 - 2nd^n/|B|$. In particular, the variety $\mathrm{V}(g_{1,s}, \dots, g_{n,s})$ is nonempty and every point is isolated. Further, this variety contains points from each component of $\mathrm{V}(g_{1,s}, \dots, g_{s,s})$. By the previous observation, it contains points from $\mathrm{V}(f_1, \dots, f_m)$. The converse direction of Lemma 4.8 now shows that $\beta_s = 0$. Using a union bound we deduce that the above algorithm succeeds with probability at least $1 - 5nd^n/|B|$, which can be lower bounded by $2/3$ using the size of B. □

4.3 Counting solutions in zero-dimensional systems

As we have seen so far, resultants are a useful tool for deciding the satisfiability of systems of polynomial equations, even in the non-square and affine cases. If a system of equations is satisfiable, a natural next task is to determine how many solutions the system has. Of course, the space of solutions may be positive-dimensional, in which case there are an infinite number of solutions. When the solution set is zero-dimensional (and hence a finite set), we can meaningfully speak about the task of counting the number of solutions. In this subsection, we show that the resultant is also useful for the task of counting the number of solutions to a zero-dimensional system of equations.

We start with the easier case of zero-dimensional projective systems.

Lemma 4.11. *Suppose that (A) and (B) hold, and let $F_1, \dots, F_m \in \mathbb{F}[x_1, \dots, x_n]$ be homogeneous polynomials of degree at most d and height at most h. Suppose the coefficient subfield of $\mathbb{F}$ has at least $100nd^{2n}$ elements. Suppose $\mathrm{V}(F_1, \dots, F_m)$ is a finite nonempty set. There is a polynomial-time Monte Carlo algorithm with success probability $2/3$ that takes as inputs $F_1, \dots, F_m$, and produces two sets of polynomials $G_{0,1}, \dots, G_{n,1}$ and $G_{0,2}, \dots, G_{n,2}$ with coefficients in $\mathbb{F}[u]$ with the following properties.*

- *Each $G_{i,j}$ is homogeneous in $x_0, \dots, x_n$ of degree at most d and has degree at most one in u.*
- *Each $G_{i,j}$ has height at most $h \cdot (n \log d)^c$ for a universal constant c.*

- *The size of* $\mathrm{V}(F_1, \ldots, F_m)$ *is exactly the number of distinct roots (in* $\overline{\mathbb{F}}$*) of*

$$\gcd(\mathrm{Res}(G_{0,1}, \ldots, G_{n,1}), \mathrm{Res}(G_{0,2}, \ldots, G_{n,2})),$$

where these resultants are computed by regarding the $G_{i,j}$ *as polynomials in* $\mathbf{x}$ *with coefficients in* $\mathbb{F}(u)$*.*

Proof. If the coefficient subfield of $\mathbb{F}$ is $\mathbb{Q}$, let B be the set of natural numbers $\{1, \ldots, 100nd^{2n}\}$. If the coefficient subfield of $\mathbb{F}$ is finite, let B be the set of nonzero elements of the coefficient subfield. We work with the polynomials

$$\left\{ x_j^{d-\deg F_i} \cdot F_i \mid i \in [m], j \in \{0, 1, \ldots, n\} \right\}$$

instead of the original polynomials, as this does not change the zero set but ensures that all polynomials in our system have the same degree. In the rest of the proof we continue to use $F_1, \ldots, F_m$ to refer to this new set of polynomials.

Let $V \coloneqq \mathrm{V}(F_1, \ldots, F_m)$ be the zero set of $F_1, \ldots, F_m$. It consists of at most d^n points by Bézout's inequality ([BCS97, Theorem 8.28]). We apply a random change of coordinates to ensure that all points in V lie on the chart $x_0 = 1$. We do this by replacing each x_i by $\sum_{j=0}^{n} \gamma_{i,j} x_j$, where $\gamma_{i,j}$ are picked from B independently and uniformly at random. With probability at least $1 - d^n/|B|$, the points in V lie on $x_0 = 1$ after this linear transformation.

Let $G_{1,1}, \ldots, G_{n,1}$ be a set of random linear combinations of the equations $F_1, \ldots, F_m$, where each coefficient is picked independently and uniformly from B. By Lemma 4.10, with probability at least $1 - 2nd^n/|B|$, the zero set $V_1 \coloneqq \mathrm{V}(G_{1,1}, \ldots, G_{n,1})$ is a finite set and contains V as a subset. The system $F_1|_{x_0=0} = \cdots = F_m|_{x_0=0} = 0$ is unsatisfiable. The system $G_{1,1}|_{x_0=0} = \cdots = G_{n,1}|_{x_0=0} = 0$ consists of random linear combinations of the $F_i|_{x_0=0}$, so by Lemma 4.10, this system has no roots with probability at least $1 - 2nd^n/|B|$. Equivalently, with this probability, every point of V_1 lies on the chart $x_0 = 1$.

Similarly, let $G_{1,2}, \ldots, G_{n,2}$ be a set of random linear combinations of $F_1, \ldots, F_m$ and let V_2 be their zero set. It again is a finite set that contains V and lies on the chart $x_0 = 1$. Further, we claim that if V_1 is a finite set, then $V_1 \cap V_2 = V$ with probability at least $1 - d^n/|B|$. If $\boldsymbol{\alpha} \in V_1 \setminus V$ is a point, then there is some F_j such that $F_j(\boldsymbol{\alpha}) \neq 0$, therefore with probability at least $1 - 1/|B|$, we have $G_{1,2}(\boldsymbol{\alpha}) \neq 0$. The fact that V_1 is finite implies that $|V_1| \leq d^n$, so the claimed bound on the probability that $V_1 \cap V_2 = V$ follows by a union bound.

We now sample $\beta_1, \ldots, \beta_n \in B$ independently and uniformly, and define $L \coloneqq x_0 + \beta_1 u x_1 + \beta_2 u x_2 + \cdots + \beta_n u x_n$. By the Poisson formula (Theorem 4.3), the resultant $\mathrm{Res}(G_{1,1}, \ldots, G_{n,1}, L)$ can be factored as

$$\mathrm{Res}(G_{1,1}, \ldots, G_{n,1}, L) = c \cdot \prod_{\boldsymbol{\alpha} \in V_1} L(\boldsymbol{\alpha})^{m_{\boldsymbol{\alpha}}},$$

where $m_{\boldsymbol{\alpha}}$ is the multiplicity of $\boldsymbol{\alpha}$ in V_1 and $c \in \mathbb{F} \setminus \{0\}$. Simplifying, and using the fact that all roots lie on the affine chart $x_0 = 1$, we obtain

$$\mathrm{Res}(G_{1,1}, \ldots, G_{n,1}, L) = c \cdot \prod_{\boldsymbol{\alpha} \in V_1} \left(1 + u \sum_{i=1}^{n} \beta_i \alpha_i\right)^{m_{\boldsymbol{\alpha}}}.$$

Since the β_i were picked randomly, the sum $\sum_{i=1}^{n} \beta_i \alpha_i$ takes distinct values for distinct roots with probability at least $1 - d^{2n}/|B|$. Similarly, we have

$$\mathrm{Res}(G_{1,2}, \ldots, G_{n,2}, L) = c' \cdot \prod_{\boldsymbol{\alpha} \in V_2} \left(1 + u \sum_{i=1}^{n} \beta_i \alpha_i\right)^{n_{\boldsymbol{\alpha}}},$$

where $n_{\boldsymbol{\alpha}}$ is the multiplicity of $\boldsymbol{\alpha}$ in V_2 and $c' \in \mathbb{F} \setminus \{0\}$. It is now clear that if we take the GCD of these two resultants, then the number of distinct roots of the GCD in $\overline{\mathbb{F}}$ is equal to the number of distinct points in $V_1 \cap V_2 = V$. Therefore we are done by setting $G_{0,1} = G_{0,2} = L$ and using a union bound to control the probabilities of all the required events. □

We now give our main reduction, showing that the resultant can be used to count the number of solutions to a zero-dimensional affine system. The idea will be similar to that in Lemma 4.11. We will take random linear combinations of the given system to reduce to the case when the number of variables and polynomials are the same. Here, we have to use the generalized characteristic polynomial instead of the resultant itself to compute the sizes of zero sets of these random systems.

Proposition 4.12. *Suppose that (A) and (B) hold, and let $f_1, \dots, f_m \in \mathbb{F}[x_1, \dots, x_n]$ be polynomials of degree at most d and height at most h. Suppose the coefficient subfield of $\mathbb{F}$ has at least $100nd^{2n}$ elements. Suppose $\mathrm{V}(f_1, \dots, f_m)$ is a finite nonempty set. There is a polynomial-time Monte Carlo algorithm with success probability $2/3$ that takes as inputs $f_1, \dots, f_m$, and produces two sets of polynomials $G_{0,1}, \dots, G_{n,1}$ and $G_{0,2}, \dots, G_{n,2}$ with coefficients in $\mathbb{F}[u, t]$ with the following properties.*

- *Each $G_{i,j}$ is homogeneous in $x_0, \dots, x_n$ of degree at most d and has degree at most one in u and t.*
- *Each $G_{i,j}$ has height at most $h \cdot (n \log d)^c$ for a universal constant c.*
- *The size of $\mathrm{V}(f_1, \dots, f_m)$ is exactly the number of distinct roots (in $\overline{\mathbb{F}}$) of*

$$\gcd(\mathrm{TP}_u \, \mathrm{TT}_t \, \mathrm{Res}(G_{0,1}, G_{1,1}, \dots, G_{n,1}), \mathrm{TP}_u \, \mathrm{TT}_t \, \mathrm{Res}(G_{0,2}, G_{1,2}, \dots, G_{n,2})),$$

  where these resultants are computed by regarding the $G_{i,j}$ as polynomials in $\mathbf{x}$ with coefficients in $\mathbb{F}(t, u)$.

Proof. If the coefficient subfield of $\mathbb{F}$ is $\mathbb{Q}$, let B be the set of natural numbers $\{1, \dots, 100nd^{2n}\}$. If the coefficient subfield of $\mathbb{F}$ is finite, let B be the set of nonzero elements of the coefficient subfield. Define $V \coloneqq \mathrm{V}(f_1, \dots, f_m)$. Let $g_{1,1}, \dots, g_{n,1}$ and $g_{1,2}, \dots, g_{n,2}$ be two random sets of linear combinations of $f_1, \dots, f_m$, with coefficients picked from B. Let $V_1 \coloneqq \mathrm{V}(g_{1,1}, \dots, g_{n,1})$ and $V_2 \coloneqq \mathrm{V}(g_{1,2}, \dots, g_{n,2})$. By Bézout's inequality and Lemma 4.10, with probability at least $1 - 6nd^n/|B|$, the zero sets V_1 and V_2 are finite and satisfy $V_1 \cap V_2 = V$ (refer to the proof of Lemma 4.11 for details).

We now define $G_{i,j} \coloneqq g_{i,j}^h + t \cdot x_i^{d_i}$, where d_i is the degree of $g_{i,j}$ and $g_{i,j}^h$ is the homogenization of $g_{i,j}$ with respect to a new variable x_0. Finally, define $L \coloneqq x_0 + \beta_1 u x_1 + \cdots + \beta_n u x_n$ where $\beta_1, \dots, \beta_n$ are sampled independently and uniformly at random from B. Consider now

$$(R_1)_0^* \coloneqq \mathrm{TT}_t \, \mathrm{Res}(G_{1,1}, \dots, G_{n,1}, L).$$

By Lemma 4.7, we can deduce that $(R_1)_0^*$ factors into a product of linear forms, one for each point in the limiting set $(V_1)_0^*$. Every point of V_1 is in this this set, and further these are the only points of $(V_1)_0^*$ that lie on the chart $x_0 = 1$. The remaining factors that correspond to points in $(V_1)_0^*$ on the hyperplane $x_0 = 0$ are just the polynomial u. Therefore, if we consider $\mathrm{TP}_u(R_1)_0^*$, every factor is of the form $(1 + u \sum_{i=1}^n \beta_i \alpha_i)$ of some $\boldsymbol{\alpha} \in V_1$, and for every $\boldsymbol{\alpha} \in V_1$ there is a factor of this form. Note that we make no claim about the multiplicities. With probability at least $1 - d^{2n}/|B|$, the coefficient of u corresponding to each point, namely $\sum_{i=1}^n \beta_i \alpha_i$, is distinct. We can do the same operation with $g_{1,2}, \dots, g_{n,2}$. Then the number of distinct roots of $\gcd(\mathrm{TP}_u(R_1)_0^*, \mathrm{TP}_u(R_2)_0^*)$ in $\overline{\mathbb{F}}$ is clearly the size of $V_1 \cap V_2$ with probability at least $1 - 2d^{2n}/|B|$ over the choice of $\beta_1, \dots, \beta_n$. Hence we are done by setting $G_{0,1} = G_{0,2} = L$. □

5 Constant-depth circuits for the resultant

In this section, we design a polylogtime-uniform family of constant-depth arithmetic circuits for the multivariate resultant $\mathrm{Res}_{d_0,\dots,d_n}$. Our circuits will be indexed by $n, d_0, \dots, d_n$, and our base field is $\mathbb{Q}$. The following subsection will establish some notation and explain the high level idea of our construction. The subsequent subsections will contain the details.

5.1 Notation and proof overview

Let n be a positive integer, and let $d_0, \dots, d_n$ be positive integers. Let $\mathbf{u}_0, \dots, \mathbf{u}_n$ be $n+1$ sets of variables, with $\mathbf{u}_i = (u_{i,\boldsymbol{\alpha}})$ for all multi-indices $\boldsymbol{\alpha} = (\alpha_0, \dots, \alpha_n)$ such that $|\boldsymbol{\alpha}| = d_i$. In other words, $\mathbf{u}_i$ consists of one variable for each monomial in $n+1$ variables of degree d_i. We let U denote the sum of the sizes of these sets of variables, so $U = \sum_{i=0}^{n} \binom{n+d_i}{d_i}$. We use S to denote the polynomial ring $\mathbb{Q}[\mathbf{u}_0, \dots, \mathbf{u}_n]$, and we write $\mathbb{K}$ for the field of fractions of S. Recall that the multivariate resultant (Section 4.1) is an element of the ring S with integer coefficients.

For $j = 0, \dots, n$, we define polynomials $F_j \in S[\mathbf{x}]$ as

$$F_j = \sum_{\boldsymbol{\alpha} \in \mathbf{u}_j} u_{j,\boldsymbol{\alpha}} x_0^{\alpha_0} \cdots x_n^{\alpha_n}.$$

Any set of forms $P_0, \dots, P_n$ of degrees $d_0, \dots, d_n$ can be obtained by specializing $F_0, \dots, F_n$. We wish to use the Poisson formula (Theorem 4.3) to compute the multivariate resultant. The formula states that the following identity holds.

$$\mathrm{Res}_{d_0,\dots,d_n}(F_0, \dots, F_n) = \mathrm{Res}_{d_0,\dots,d_{n-1}}(\overline{F}_0, \dots, \overline{F}_{n-1})^{d_n} \cdot \prod_{\boldsymbol{\alpha} \in \mathrm{V}(f_0,\dots,f_{n-1})} f_n(\boldsymbol{\alpha})^{m(\boldsymbol{\alpha})},$$

where $\overline{F}_j = F_j|_{x_n=0}$ and $f_j = F_j|_{x_n=1}$. We will apply this formula recursively to compute the first term, therefore we will have to study the polynomials $F_j|_{x_{i+1}=0,\dots,x_n=0}$ and $F_j|_{x_i=1,x_{i+1}=0,\dots,x_n=0}$ for every i. We give these polynomials and some related objects names to be able to refer to them easily.

For any $i, j \in \{0, 1, \dots, n\}$, we let $\mathbf{u}_{i,j} \subseteq \mathbf{u}_i$ denote those variables $u_{i,\boldsymbol{\alpha}}$ with $\alpha_{j+1} = \cdots = \alpha_n = 0$. In other words, these variables $\mathbf{u}_{i,j}$ correspond to monomials that only depend on $x_0, \dots, x_j$. We let $F_{i,j}$ denote the polynomial

$$F_{i,j} = \sum_{\boldsymbol{\alpha} \in \mathbf{u}_{i,j}} u_{i,\boldsymbol{\alpha}} x_0^{\alpha_0} \cdots x_n^{\alpha_n}.$$

The polynomial $F_{i,j}$ is a generic degree-d_i form in the variables $x_0, \dots, x_j$. We use $f_{i,j}$ to denote the specialization $F_{i,j}(x_0, \dots, x_{j-1}, 1)$.

Let $\mathbf{F}_j$ denote the vector of polynomials $(F_{0,j}, \dots, F_{j,j})$. This is a vector of $j+1$ generic polynomials in $j+1$ variables of degrees $d_0, \dots, d_j$. We write U_j for the number of coefficients in this system, so $U_j = \sum_{i=0}^{j} \binom{d_i+j}{j}$.

Let $\mathrm{Res}_j \in \mathbb{Q}[\mathbf{u}_{0,j}, \dots, \mathbf{u}_{j,j}]$ denote the resultant $\mathrm{Res}_{d_0,\dots,d_j}((F_{0,j}, \dots, F_{j,j}))$ (the degrees $d_0, \dots, d_j$ are fixed throughout). The polynomial Res_j depends on the degrees $d_0, \dots, d_j$, but we suppress this from notation to avoid clutter. Given a set of forms $P_0, \dots, P_j \in \mathbb{Q}[x_0, \dots, x_j]$ of degrees $d_0, \dots, d_j$, we use $\mathrm{Res}_j(\mathbf{P})$ to denote the evaluation of Res_j at the coefficients of $P_0, \dots, P_j$.

In the notation above, the second term in the Poisson formula involves the roots of the polynomials $f_{0,n}, \dots, f_{n-1,n}$. These roots lie in the algebraic closure of $\mathbb{K}$. However, it is computationally difficult to describe and perform arithmetic with elements of $\overline{\mathbb{K}}$. We will instead work with roots that lie in a power series ring. These are easier to manipulate, since we usually only require computing them to some bounded precision.

The roots of $f_{0,n}, \ldots, f_{n-1,n}$ do not admit power series representations in the ring $\mathbb{Q}[\![\mathbf{u}_0, \ldots, \mathbf{u}_n]\!]$. This is already apparent even in the case when $n = 1$, and $d_0 = 2$. In this case we want the roots of a generic bivariate polynomial $u_{0,(2,0)}x_0^2 + u_{0,(1,1)}x_0 + u_{0,(0,2)}$. The expression for the roots involves the square root of the discriminant $u_{0,(1,1)}^2 - 4u_{0,(2,0)}u_{0,(0,2)}$, but this element is not a square in the ring $\mathbb{Q}[\![\mathbf{u}_0, \ldots, \mathbf{u}_n]\!]$.

We will instead find power series roots of $f_{0,n}, \ldots, f_{n-1,n}$ using a homotopy. We introduce a new variable t, and for each $i = 0, \ldots, n$ we define polynomials

$$H_{i,n} := (1-t)G_{i,n} + tF_{i,n},$$

where $G_{i,n} \in \mathbb{Q}[x_0, \ldots, x_n]$ are yet to be defined. We have $H_{i,n}|_{t=1} = F_{i,n}$, and $H_{i,n}|_{t=0} = G_{i,n}$. The polynomials $H_{0,n}, \ldots, H_{n,n}$ define a homotopy between the roots of $F_{0,n}, \ldots, F_{n,n}$ and the roots of $G_{0,n}, \ldots, G_{n,n}$. The system of equations defined by the polynomials $G_{0,n}, \ldots, G_{n,n}$ is called the *initial system* the homotopy. We define $h_{i,n} := H_{i,n}|_{x_n=1}$ and $g_{i,n} := G_i|_{x_n=1}$. The forms $G_{i,n}$ will be chosen such that computing the roots of the equations $g_{0,n}, \ldots, g_{n-1,n}$ can be easily performed by constant-free uniform constant-depth circuits. We will also ensure that all roots of the system are simple, and that the Jacobian at each root can also be computed by such circuits.

A consequence of the simplicity of the roots of $g_{0,n}, \ldots, g_{n-1,n}$ is that $h_{0,n}, \ldots, h_{n-1,n}$ admit power series roots in t, whose constant terms are exactly the roots of $g_{0,n}, \ldots, g_{n-1,n}$. Hensel lifting and Newton iteration give constructive proofs of the existence of these roots. In fact a Newton iteration with linear rate of convergence gives a constructive proof of the existence of roots of the above system in $S[\![t]\!]$, that is, roots that are power series in one variable with coefficients are polynomial in the remaining variables. While Newton iteration and Hensel lifting give constructive proofs, following either construction only gives us uniform circuits of polylogarithmic depth for these roots. This is insufficient for our applications. We will instead compute these roots using an explicit implicit function theorem ([AY83, Proposition 20.3]).

More generally, to handle the recursive terms in the Poisson formula, we define polynomials $G_{i,j}, H_{i,j}$ for all $1 \leq j \leq n$ and $0 \leq i \leq j$ by specializing $G_{0,n}, \ldots, G_{n,n}$ and $H_{0,n}, \ldots, H_{n,n}$ respectively. In each case, $G_{0,j}, \ldots, G_{j,j}$ will be the initial system for $H_{0,j}, \ldots, H_{j,j}$. We will apply the Poisson formula and the above mentioned explicit formulas for the roots to compute the resultant of $H_{0,n}, \ldots, H_{n,n}$. This computation will be carried out in $S[\![t]\!]$ up to a certain precision. This precision will be chosen such that truncating and setting $t = 1$ will allow us to recover $\mathrm{Res}_n(\mathbf{F}_n)$.

The next subsection describes the initial systems, and shows that their roots, and the Jacobian evaluated at the roots can be computed by uniform constant-depth circuits. The subsection after that carries out the rest of the discussion above.

We point out that the algorithm in [JS07] already combines homotopy techniques with the Poisson formula to compute resultants. The homotopy in that article uses starting systems with a similar structure as the polynomials $g_{i,j}$ mentioned above, but bypasses the introduction of the new variable t. If we let $\boldsymbol{\nu}_0, \ldots, \boldsymbol{\nu}_j$ be the coefficients of these polynomials, their algorithm uses Newton iteration to produce a straight-line program that computes rational function approximations to the roots of $f_{0,j}, \ldots, f_{j-1,j}$ in $\mathbb{Q}[[\mathbf{u}_0 - \boldsymbol{\nu}_0, \ldots, \mathbf{u}_{j-1} - \boldsymbol{\nu}_{j-1}]]$, injects them into the product formula, and eliminates divisions. The explicit implicit function theorem in [AY83] also applies for the homotopy in [JS07], but in this context, it would result in circuits of double exponential size.

5.2 The initial systems

We now describe the initial systems, that is, the polynomials $G_{i,j}$. The following construction is from [HJSS02] (and is also briefly mentioned in [MSW95]), in the more general case of multi-homogeneous systems.

Definition 5.1 (Initial System)**.** For any integer m and for $j \in \{0, 1, \ldots, n\}$, we define the linear form $L_{j,m}$ as

$$L_{j,m}(\mathbf{x}) \coloneqq x_0 + mx_1 + \cdots + m^{j-1}x_{j-1} + m^j x_j.$$

We use $\ell_{j,m}$ to denote the specialization of $L_{j,m}$ at $x_j = 1$.

Define subsets of integers $A_0, \ldots, A_n$ as $A_0 \coloneqq \{1, \ldots, d_0\}$, $A_1 \coloneqq \{d_0 + 1, \ldots, d_0 + d_1\}$, and so on. The set A_i has size d_i, and all of these sets are pairwise disjoint. For $i \in \{0, 1, \ldots, n\}$, we define the form $G_{i,j} \in \mathbb{Q}[\mathbf{x}]$ as

$$G_{i,j}(\mathbf{x}) \coloneqq \prod_{k \in A_i} L_{j,k}(\mathbf{x}).$$

We use $\mathbf{G}_j$ to denote the vector of polynomials $(G_{0,j}, \ldots, G_{j,j})$. Finally, $g_{i,j}$ denotes the specialization of $G_{i,j}$ to $x_j = 1$, and $\mathbf{g}_j$ denotes the vector of polynomials $(g_{0,j}, \ldots, g_{j-1,j})$. ◊

Observe that each $G_{i,j}$ is a form of degree d_i in the variables $x_0, \ldots, x_j$. The reason why we use $\mathbf{g}_j$ to denote $g_{0,j}, \ldots, g_{j-1,j}$ (and not the more natural choice of $g_{0,j}, \ldots, g_{j,j}$) is that when applying the Poisson formula to $H_{0,j}, \ldots, H_{j,j}$, the term corresponding to the product of roots will only involve solving the system $H_{0,j}|_{x_j=1}, \ldots, H_{j-1,j}|_{x_j=1}$.

The following lemma captures some basic properties about the zeroes of the systems $\mathbf{g}_j$. Define $e_0 \coloneqq 0$. For $i = 1, \ldots, n$ define $e_i \coloneqq d_0 + \cdots + d_{i-1}$.

Lemma 5.2. *For each $j \in \{0, 1, \ldots, n\}$, let $B_j \coloneqq [d_0] \times [d_1] \times \cdots \times [d_{j-1}]$. For each $\mathbf{c} \in B_j$, the linear system $\ell_{j,c_0} = \ell_{j,e_1+c_1} = \cdots = \ell_{j,e_{j-1}+c_{j-1}} = 0$ has a single common solution, which we denote by $\mathbf{r}_\mathbf{c}$. The set of such $\mathbf{r}_\mathbf{c}$ are exactly the set of common zeroes of $\mathbf{g}_j$.*

Proof. For each choice of $\mathbf{c}$, the matrix corresponding to the linear system is a Vandermonde matrix, and is therefore invertible. This shows that a unique solution exists. Further, for any fixed $\mathbf{c}$, if ℓ is a linear form of the type $\ell_{j,m}$ for any m that is not among $e_0 + c_0, \ldots, e_{j-1} + c_{j-1}$, then the system of linear equations

$$\ell, \ell_{j,e_0+c_0}, \ldots, \ell_{j,e_{j-1}+c_{j-1}}$$

is unsatisfiable. This can be seen by homogenizing the system and observing that the corresponding matrix is again a Vandermonde matrix, so the zero vector is the only solution. These two facts also imply that the set of common zeroes of $\mathbf{g}_j$ are exactly $\mathbf{r}_\mathbf{c}$. □

Next, we compute the Jacobian of the system $\mathbf{g}_j$ at each of its solutions $\mathbf{r}_\mathbf{c}$.

Lemma 5.3. *Let the notion be as in the statement of Lemma 5.2. For $\mathbf{c} \in B_j$, and $i \le j$, define $\kappa_{\mathbf{c},i}$ as*

$$\kappa_{\mathbf{c},i} \coloneqq \frac{g_{i,j}}{\ell_{j,e_i+c_i}}(\mathbf{r}_\mathbf{c}).$$

Then the Jacobian of $\mathbf{g}_j$ at $\mathbf{r}_\mathbf{c}$, denoted $\mathcal{J}_\mathbf{c}$, is an invertible matrix and factors as

$$\mathcal{J}_\mathbf{c} = \operatorname{diag}(\kappa_{\mathbf{c},0}, \ldots, \kappa_{\mathbf{c},j-1}) \cdot \operatorname{Vand}(e_0 + c_0, \ldots, e_{j-1} + c_{j-1}).$$

Proof. The partial derivative of $g_{i,j}$ with respect to x_a is given by

$$\partial_a g_{i,j}(\mathbf{x}) = \sum_{k'=1}^{d_i} \prod_{k \neq k'} \ell_{j,e_i+k}(\mathbf{x}) \cdot \partial_a \ell_{j,e_i+k'}(\mathbf{x}) = \sum_{k'=1}^{d_i} \prod_{k \neq k'} \ell_{j,e_i+k}(\mathbf{x}) \cdot \left(e_i + k'\right)^a.$$

When evaluated at $\mathbf{r}_\mathbf{c}$, the only term that survives in the above summation is the one where ℓ_{j,e_i+c_i} is omitted, since ℓ_{j,e_i+c_i} vanishes at $\mathbf{r}_\mathbf{c}$. In this summand, the first product term when evaluated at

$\mathbf{r_c}$ is exactly $\kappa_{\mathbf{c},i}$. This shows the claimed factorization for the Jacobian. The proof of Lemma 5.2 shows that the $\kappa_{\mathbf{c},i}$ are nonzero, since $\mathbf{r_c}$ is not a root of any linear polynomial that is a factor of $g_{i,j}$, other than ℓ_{j,e_i+c_i}. The fact that the Jacobian is invertible now follows from the factorization. □

For each j, the solution $\mathbf{r_c}$, the Jacobian $\mathcal{J}_\mathbf{c}$, and its inverse $\mathcal{J}_\mathbf{c}^{-1}$ are all rational functions of $\mathbf{c}$. The following lemma describes polylogtime-uniform, constant-free, constant-depth circuits that take as input $\mathbf{c}$ and compute these rational functions.

Lemma 5.4. *There exists a polylogtime-uniform family of constant-free circuits $\mathcal{C}^{\text{init}}$ indexed by $j, d_0, \ldots, d_{j-1}$ with the following properties.*

- *The circuit $C^{\text{init}}_{j,d_0,\ldots,d_{j-1}}$ has size polynomial in $j + d_0 + \cdots + d_{j-1}$, and depth bounded by a universal constant.*
- *The circuit $C^{\text{init}}_{j,d_0,\ldots,d_{j-1}}$ has j input gates, labeled by the variables $z_0, \ldots, z_{j-1}$. The circuit has $j + 2j^2$ output gates.*
- *On input $(c_0, \ldots, c_{j-1}) \in [d_0] \times \cdots \times [d_{j-1}]$, this circuit computes $\mathbf{r_c}$, the entries of the Jacobian $\mathcal{J}_\mathbf{c}$, and the entries of its inverse $\mathcal{J}_\mathbf{c}^{-1}$.*

Proof. We first describe the circuit $C = C^{\text{init}}_{j,d_0,\ldots,d_{j-1}}$. Recall that e_k denotes the sum $d_0 + \cdots + d_{k-1}$. The circuit C has gates that compute all the integers between 1 and e_j by repeatedly adding the constant 1. It has + gates that compute the polynomials $e_0 + z_0, \ldots, e_{j-1} + z_{j-1}$ by adding z_j to the integer e_{j-1}. The circuit has gates that compute the polynomials $-(e_0 + z_0)^j, \ldots, -(e_{j-1} + z_{j-1})^j$. This will require introducing more copies of the gates that compute $e_0 + z_0$. The gates for these computations will be named $(1, *)$ for some $*$ (which itself will be a tuple), analogous to how gates performing similar tasks were named in the proof of Lemma 3.1. The circuit has a copy of the circuit C' for the inverse of a general Vandermonde matrix of size $j \times j$ (Lemma 3.6), whose inputs are connected to the gates computing the above polynomials. These gates will be named $(2, v)$ where v is the name of the gate in the circuit C'. The output of this subcircuit computes the inverse of the Vandermonde matrix $\text{Vand}(e_0 + z_0, \ldots, e_{j-1} + z_{j-1})$. The circuit then multiplies the inverse of the Vandermonde matrix with the vector $(-(e_0 + z_0)^j, \ldots, -(e_{j-1} + z_{j-1})^j)$. Denote the entries of the resulting vector by $\mathbf{r_z}$. These gates will be named $(3, *)$.

Next, for each $i \in \{0, \ldots, j-1\}$, the circuit has gates that compute the polynomials

$$\sum_{k'=1}^{d_i} \prod_{k \neq k'} \ell_{j,e_i+k}(x_0, \ldots, x_{j-1}),$$

and evaluates them at $\mathbf{r_z}$. Call these evaluations $\kappa_{\mathbf{z},i}$, the gates will be named $(4, *)$. The circuit then has gates that compute the matrix product

$$\mathcal{J}_\mathbf{z} \coloneqq \text{diag}(\kappa_{\mathbf{z},1}, \ldots, \kappa_{\mathbf{z},j-1}) \cdot \text{Vand}(e_0 + z_0, \ldots, e_{j-1} + z_{j-1})$$

Finally, the circuit has gates that compute the inverse of $\mathcal{J}_\mathbf{z}$ using the factorization above and the entries of $\text{Vand}(e_0 + z_0, \ldots, e_{j-1} + z_{j-1})^{-1}$ computed earlier. These gates can also be named as above. In all cases, a natural numbering of the predecessor of each gates is clear from the description.

We argue that evaluating this circuit on $\mathbf{c}$ computes the root $\mathbf{r_c}$, the Jacobian $\mathcal{J}_\mathbf{c}$ and its inverse correctly. It follows from the definition of $\mathbf{r_c}$ that $\mathbf{r_z}(\mathbf{c}) = \mathbf{r_c}$. To see that $\kappa_{\mathbf{z},i}(\mathbf{c}) = \kappa_{\mathbf{c},i}$, we observe that when evaluated at $\mathbf{r_c}$, the only summand in the expression for $\kappa_{\mathbf{z},i}$ that does not vanish is the one where ℓ_{j,e_i+c_i} is omitted. From this and Lemma 5.3, it follows that the inverse and Jacobian are computed correctly.

The claims on the size and depth of the circuit follow from the above construction and the bounds on the sizes of the subcircuits used. Uniformity follows by a simulation argument, as in the proof of Lemma 3.2, together with the above description. For the gates in the copy of C', that is, gates named $(2, v)$ for some v, since v can be deduced from this name, the Turing machine can simulate the machine that decides the direct connection language of the circuit family from Lemma 3.6 whenever required. For all other gates, the above description of the circuit can be used to design the Turing machine, similar to the proof of Lemma 3.1. □

5.3 The resultant of the system defining the homotopy

We can now describe the system that defines the homotopy.

Definition 5.5. For any $i, j \in \{0, 1, \ldots, n\}$, let $H_{i,j}(\mathbf{x}, t)$ denote the polynomial

$$H_{i,j}(\mathbf{x}, t) \coloneqq (1-t)G_{i,j}(\mathbf{x}) + tF_{i,j}(\mathbf{x}),$$

which we regard as an element of $\mathbb{K}(t)[\mathbf{x}]$. For any such i and j, we use $h_{i,j}$ to denote the specialization of $H_{i,j}$ at $x_j = 1$. Let $\mathbf{H}_j$ denote the vector of polynomials $(H_{0,j}, \ldots, H_{j,j})$ and $\mathbf{h}_j$ denote the vector of polynomials $(h_{0,j}, \ldots, h_{j-1,j})$. ◊

Specializing $H_{i,j}$ and $h_{i,j}$ to $t = 0$ gives us $G_{i,j}$ and $g_{i,j}$ respectively, and specializing to $t = 1$ gives $F_{i,j}$ and $f_{i,j}$. Further, for $j \in [n]$, setting x_j to zero in $H_{0,j}, \ldots, H_{j-1,j}$ gives us $H_{0,j-1}, \ldots, H_{j-1,j-1}$. Via the Poisson formula, we can derive an expression for $\mathrm{Res}_n(\mathbf{H}_n)$.

Lemma 5.6. *For each $j \in \{0, 1, \ldots, n\}$, let $V_j \subseteq \overline{\mathbb{K}(t)}^j$ denote the set of common zeroes of $h_{0,j}, \ldots, h_{j-1,j}$. We have*

$$\mathrm{Res}_n(\mathbf{H}_n) = \left(1 + t\left(u_{0,(d_0,0,\ldots,0)} - 1\right)\right)^{\prod_{i=1}^n d_i} \times \left(\prod_{\boldsymbol{\rho} \in V_1} h_{1,1}(\boldsymbol{\rho})\right)^{\prod_{i=2}^n d_i} \times \left(\prod_{\boldsymbol{\rho} \in V_2} h_{2,2}(\boldsymbol{\rho})\right)^{\prod_{i=3}^n d_i} \times \cdots \times \left(\prod_{\boldsymbol{\rho} \in V_n} h_{n,n}(\boldsymbol{\rho})\right).$$

Proof. We have $F_{0,0} = u_{0,(d_0,0,\ldots,0)} x_0^{d_0}$ and $G_{0,0} = x_0^{d_0}$. Therefore, $\mathrm{Res}(H_{0,0}) = 1 + t\left(u_{0,(d_0,0,\ldots,0)} - 1\right)$, using the fact that the resultant of a single form in one variable is just the coefficient ([Jou91, Example 2.1]). For the systems $\mathbf{H}_j$, the Poisson formula (Theorem 4.3) applies, since the specialization to $t = 1$ results in generic forms. Unrolling the recurrence of the Poisson formula gives us the above expression. □

As described above, the sets V_j are finite sets of points with coordinates in $\overline{\mathbb{K}(t)}$. In this form, they are not amenable to manipulation by arithmetic circuits and Turing machines. Define $\mathbb{A} \coloneqq \mathbb{Q}[\mathbf{u}_0, \ldots, \mathbf{u}_n][\![t]\!]$. Consider the $h_{i,j}$ as polynomials in the ring $\mathbb{A}[x_0, \ldots, x_{j-1}]$. The above observations, combined with the discussion in Section 5.2, allows us to apply Hensel's lemma to this system of equations.

Lemma 5.7. *Let $j \in [n]$. As elements of $\mathbb{A}[x_0, \ldots, x_{j-1}]$, the set of polynomials $\mathbf{h}_j$ admits $d_0 \cdots d_{j-1}$ roots in $\mathbb{A}$. For each $\mathbf{r_c} \in B_j$ as defined in Lemma 5.2, there is a root $\boldsymbol{\rho}_\mathbf{c} \in \mathbb{A}^j$ of $\mathbf{h}_j$ that satisfies $\boldsymbol{\rho}_\mathbf{c} \pmod t = \mathbf{r_c}$. The set of polynomials has no other roots in any extension of $\mathbb{A}$.*

Proof. The first statement follows from a multivariate version of Hensel's lemma, see for example [Eis13, Exercise 7.26]. The final statement follows from Bézout's inequality. □

We will use an explicit version of the implicit function theorem to compute the $\boldsymbol{\rho}_{\mathbf{c}}$ to the required precision. To apply this theorem, we have to perform some shifting and scaling to ensure that the root $\boldsymbol{\rho}_{\mathbf{c}}$ we are computing has no constant term, and that the Jacobian at the origin after shifting is the identity.

Definition 5.8. Let $j \in [n]$ and $\mathbf{c} \in B_j$. Define the shifted and scaled system $\widetilde{\mathbf{h}}_{\mathbf{c}}$ as

$$\widetilde{\mathbf{h}}_{\mathbf{c}}(\mathbf{x}) \coloneqq \mathcal{J}_{\mathbf{c}}^{-1}\mathbf{h}_j(\mathbf{x} + \mathbf{r}_{\mathbf{c}}).$$

This system has $\boldsymbol{\rho}_{\mathbf{c}} - \mathbf{r}_{\mathbf{c}}$ as a root, which has a vanishing constant term in every coordinate. The Jacobian at zero of this system is the identity matrix. Let $\Delta_{\mathbf{c}}$ denote the Jacobian determinant of this system. Note that we hide j in the notation $\widetilde{\mathbf{h}}_{\mathbf{c}}$ for brevity, as it will always be clear from context. ◊

Each polynomial in $\widetilde{\mathbf{h}}_{\mathbf{c}}$ lies in $S[t, x_0, \ldots, x_{j-1}]$. We now quote an explicit version of the implicit function theorem, specialized to the system described in Definition 5.8.

Lemma 5.9 (Implicit Function Theorem [AY83, Proposition 20.3])**.** *Let $\widetilde{\boldsymbol{\rho}}_{\mathbf{c}} = (\widetilde{\rho}_{\mathbf{c},0}(t), \ldots, \widetilde{\rho}_{\mathbf{c},j-1}(t))$ be the root of $\widetilde{\mathbf{h}}_{\mathbf{c}}$ with vanishing constant term in every coordinate. Write $\widetilde{\rho}_{\mathbf{c},i}(t) = \sum_{N\geq 0} \widetilde{\rho}_{\mathbf{c},i,N} t^N$ with coefficients $\widetilde{\rho}_{\mathbf{c},i,N} \in S$. Then the coefficient $\widetilde{\rho}_{\mathbf{c},i,N}$ is given by*

$$\widetilde{\rho}_{\mathbf{c},i,N} = \sum_{\substack{(b_0,\ldots,b_{j-1})\in\mathbb{N}^j \\ |\mathbf{b}|\leq 2N}} (-1)^{|\mathbf{b}|} \operatorname{coeff}(x_i(\widetilde{h}_{\boldsymbol{c},0}(\mathbf{x}) - x_0)^{b_0} \cdots (\widetilde{h}_{\boldsymbol{c},j-1}(\mathbf{x}) - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{c}}, t^N x_0^{b_0} \cdots x_{j-1}^{b_{j-1}}).$$

Proof. The result we cite from [AY83, Proposition 20.3] is complex-analytic, but deriving the formal statement above from it is straightforward.

In what follows, j and $\mathbf{c}$ are fixed. Lemma 5.7 pointed out that the coefficients of $\widetilde{\rho}_{\mathbf{c}}$ are polynomials in $\mathbf{u}_0, \ldots, \mathbf{u}_n$, that is, $\widetilde{\rho}_{\mathbf{c},i,N}$ is in $\mathbb{Q}[\mathbf{u}_0, \ldots, \mathbf{u}_n]$ for all i, N. We use $Q_{i,N}$ to denote the polynomials appearing as right-hand sides in the statement of Lemma 5.9, so our claim is that $\widetilde{\rho}_{\mathbf{c},i,N} = Q_{i,N}$ for all i and all N (these polynomials actually only involve $\mathbf{u}_0, \ldots, \mathbf{u}_{j-1}$, but this has no bearing on the proof).

Choose $\boldsymbol{\nu}_0, \ldots, \boldsymbol{\nu}_n$ with entries in $\mathbb{C}$ and let $\widetilde{\mathbf{h}}_{\mathbf{c},\boldsymbol{\nu}}$ be the polynomials in $\mathbb{C}[t][x_0, \ldots, x_{j-1}]$ obtained by evaluating $\mathbf{u}$ at $\boldsymbol{\nu}$ in $\widetilde{\mathbf{h}}_{\mathbf{c}}$. Through this evaluation, we see that the unique formal power series root with vanishing constant term to $\widetilde{\mathbf{h}}_{\mathbf{c},\boldsymbol{\nu}} = 0$ is $\widetilde{\rho}_{\mathbf{c}}(\boldsymbol{\nu}, t)$, whose coefficients are $\widetilde{\rho}_{\mathbf{c},i,N}(\boldsymbol{\nu})$, for $i = 0, \ldots, j-1$ and $N \geq 0$. On the other hand, applying [AY83, Proposition 20.3] to $\widetilde{\mathbf{h}}_{\mathbf{c},\boldsymbol{\nu}}$ tells us that the ith coordinate of the unique analytic root to these equations with vanishing constant term is

$$\begin{aligned}
&\sum_{N\geq 0} \sum_{\substack{|\mathbf{B}|\in\mathbb{N}^j \\ |\mathbf{b}|\leq 2N}} (-1)^{|\mathbf{b}|} \operatorname{coeff}(x_i(\widetilde{h}_{\boldsymbol{c},\boldsymbol{\nu},0}(\mathbf{x}) - x_0)^{b_0} \cdots (\widetilde{h}_{\boldsymbol{c},\boldsymbol{\nu},j-1}(\mathbf{x}) - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{c},\boldsymbol{\nu}}, t^N x_0^{b_0} \cdots x_{j-1}^{b_{j-1}}) t^N \\
&= \sum_{N\geq 0} Q_{i,N}(\boldsymbol{\nu}) t^N,
\end{aligned}$$

where $\Delta_{\mathbf{c},\boldsymbol{\nu}}$ is the Jacobian determinant of $\widetilde{\mathbf{h}}_{\mathbf{c},\boldsymbol{\nu}}$ with respect to $x_0, \ldots, x_{j-1}$ (we note that [AY83] uses the notation $D_{w,z}^{\alpha,\beta}$ to denote the partial derivative $\partial_w^\alpha \partial_z^\beta$).

Since an analytic root is also a root over the ring of formal power series, we deduce that $\widetilde{\rho}_{\mathbf{c},i,N}(\boldsymbol{\nu}) = Q_{i,N}(\boldsymbol{\nu})$ for all i, N, and $\boldsymbol{\nu}$, so that $\widetilde{\rho}_{\mathbf{c},i,N} = Q_{i,N}$ for all i and N. □

The root $\boldsymbol{\rho}_{\mathbf{c}}$ of $\mathbf{h}_j$ is a shift of $\widetilde{\boldsymbol{\rho}}_{\mathbf{c}}$ by $\mathbf{r}_{\mathbf{c}}$. If $\rho_{\mathbf{c},i,N}$ are the coefficients of t in the coordinates of $\boldsymbol{\rho}_{\mathbf{c}}$ then $\rho_{\mathbf{c},i,N} = \widetilde{\rho}_{\mathbf{c},i,N}$ for $N \geq 1$, and $\rho_{\mathbf{c},0,N} = \mathbf{r}_{\mathbf{c}}$. Using Lemma 5.9, we can thus compute $\boldsymbol{\rho}_{\mathbf{c}}$ up to any precision of our choice. To compute the resultant, we will require the coefficients $\rho_{\mathbf{c},i,N}$ for N up to $P \coloneqq \sum_{i=0}^{n} \prod_{j \neq i} d_j$, which is the degree of $\mathrm{Res}_n(\mathbf{H}_n)$ in the variable t (Lemma 4.2). We now describe the circuits carrying out the above computations. That there exist constant-depth circuits that do the above computations follows directly from the above formula. The part that takes some work is establishing that there exist uniform circuit families that implement the expression in Lemma 5.9.

We start by showing that the shifted and scaled system, along with its Jacobian determinant, can be computed efficiently.

Lemma 5.10. *There exists a polylogtime-uniform family of constant-free circuits $\mathcal{C}^{\mathrm{scaled}}$ indexed by $j, d_0, \ldots, d_{j-1}$ with the following properties.*

- *The circuit $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$ has size polynomial in U_j and $\exp(\mathrm{poly}(j))$, and depth bounded by a universal constant.*
- *The circuit $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$ has $U_j + 2j + 1$ input gates, labeled by variables $\mathbf{u}_0, \ldots, \mathbf{u}_j$, $x_0, \ldots, x_{j-1}$, $z_0, \ldots, z_{j-1}, t$. The circuit has $j+1$ output gates. The circuit is division-free with respect to $\mathbf{u}_0, \ldots, \mathbf{u}_j, x_0, \ldots, x_{j-1}, t$.*
- *When $z_0, \ldots, z_{c-1}$ are specialized to $(c_0, \ldots, c_{j-1}) \in B_j$, the circuit computes the polynomials $\widetilde{\mathbf{h}}_{\mathbf{c}}(x_0, \ldots, x_{j-1}, t)$ and the Jacobian determinant $\Delta_{\mathbf{c}}(x_0, \ldots, x_{j-1}, t)$.*

Proof. The construction of $\mathcal{C}^{\mathrm{scaled}}$ requires us to define auxiliary circuit families $\mathcal{C}'$ and $\mathcal{C}''$. We start with the construction of $\mathcal{C}'$.

The family $\mathcal{C}'$ is also indexed by $j, d_0, \ldots, d_{j-1}$ and takes the same input variables as $\mathcal{C}$. We describe $C' = C'_{j,d_0,\ldots,d_{j-1}}$. It contains as a subcircuit a copy of the circuit $C^{\mathrm{init}}_{j,d_0,\ldots,d_{j-1}}$, constructed in Lemma 5.4, that takes as input the variables $\mathbf{z}$. Let $\mathbf{r}_{\mathbf{z}}$, $\mathcal{J}_{\mathbf{z}}$, and $\mathcal{J}_{\mathbf{z}}^{-1}$ denote the outputs of this subcircuit. The circuit C' also contains a subcircuit that computes the polynomials $\mathbf{h}_j$. Computing the polynomials $\mathbf{h}_j$ amounts to computing the polynomials $\mathbf{f}_j$ and $\mathbf{g}_j$, and can be done using the definition of $\mathbf{g}_j$. The subcircuit that computes $\mathbf{h}_j$ is given as input $\mathbf{x} + \mathbf{r}_{\mathbf{z}}$, so it computes $\mathbf{h}_j(\mathbf{x} + \mathbf{r}_{\mathbf{z}})$. Next, C' computes the matrix-vector product of $\mathcal{J}_{\mathbf{z}}^{-1}$ and $\mathbf{h}_j(\mathbf{x} + \mathbf{r}_{\mathbf{z}})$. Denote the resulting output by $\widetilde{\mathbf{h}}_{\mathbf{z}}$. Note that if we specialize $\mathbf{z}$ to $\mathbf{c} \in B_j$, then $\widetilde{\mathbf{h}}_{\mathbf{z}}$ is exactly $\widetilde{\mathbf{h}}_{\mathbf{c}}$. It is straightforward to see that the family $\mathcal{C}'$ described above is polylogtime-uniform, using a simulation argument as in the proof of Lemma 3.2. The size of C' is polynomial in U_j, and the degree in $\mathbf{x}$ of each output is bounded by $d_0 + \cdots + d_{j-1}$. The depth of C' is bounded by a universal constant. Further, the denominators in all division gates are free of any variable in $\mathbf{x}$.

We now construct $\mathcal{C}''$. The family $\mathcal{C}''$ is also indexed by $j, d_0, \ldots, d_{j-1}$ and takes the same input variables as $\mathcal{C}$. It is obtained by applying Lemma 3.4 to the family $\mathcal{C}'$, with distinguished variables $x_0, \ldots, x_{j-1}$. The statements at the end of the previous paragraph show that $\mathcal{C}'$ meets all the required assumptions for this interpolation result to apply. The resulting circuit $C'' = C''_{j,d_0,\ldots,d_{j-1}}$ has size polynomial in U_j, and computes the coefficients of every output of $C'_{j,d_0,\ldots,d_{j-1}}$ in each of the variables $x_0, \ldots, x_{j-1}$. This circuit family is also polylogtime-uniform and consists of constant-depth circuits.

We now describe the circuit $C = C^{\mathrm{scaled}}_{n,j,d_0,\ldots,d_{j-1}}$. The circuit C contains a copy of the circuit $C' = C'_{j,d_0,\ldots,d_{j-1}}$, whose outputs are also outputs of C. These gates will be named $(1, v)$ where v is the name of the gate within C'. Next C contains a copy of $C'' = C''_{j,d_0,\ldots,d_{j-1}}$. These gates will be named $(2, v)$ where v is the name of the gate within the circuit obtained from interpolation.

Using the coefficients of x_k in $\widetilde{\mathbf{h}}_{\mathbf{z}}$ computed in the subcircuit C'', the circuit C then computes the entries of the Jacobian matrix of $\widetilde{\mathbf{h}}_{\mathbf{z}}$. The partial derivatives with respect to x_k are just linear combinations, weighted by powers of x_k, of the coefficients of $\widetilde{\mathbf{h}}_{\mathbf{z}}$ viewed as a polynomial in x_k. Finally, the circuit C computes the determinant of this Jacobian using the trivial depth-two circuit for determinants of $j \times j$ matrices, that is, by expanding it out into $j!$ summands. The gates for these computations will be named $(3, *)$.

We now bound the size of C. The size of the subcircuits C', C'', and the circuitry to compute the partial derivatives can be bounded by $\mathrm{poly}(U_j)$. To compute the determinant, we use a depth-two circuit that has size polynomial in $j!$, which can be bounded by $\exp(\mathrm{poly}(j))$. The size of C is therefore polynomial in U_j and $\exp(\mathrm{poly}(j))$ as claimed. The proof of uniformity is a simulation argument, just as the previous proofs. As before, our naming scheme allows us to deduce the name of a gate within each of the subcircuits, which can be used to simulate the Turing machine that decides the direct connection language of each subcircuit whenever required. □

Using Lemma 5.10, we now show that the truncations of the power series roots can be computed by uniform constant-depth circuits.

Lemma 5.11. *There exists a polylogtime-uniform family of constant-free circuits $\mathcal{C}^{\mathrm{root}}$ indexed by parameters $n, j, d_0, \ldots, d_n$ with the following properties.*

- *The circuit $C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$ has size bounded by $P^{\mathrm{poly}(n)}$, where $P \coloneqq \sum_{i=0}^{n} \prod_{j \neq i} d_j$, and depth bounded by a universal constant.*
- *The circuit $C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$ has $U_j + j + 1$ input gates, labeled by variables $\mathbf{u}_0, \ldots, \mathbf{u}_{j-1}, z_0, \ldots, z_{j-1}, t$. The circuit has j output gates. The circuit is division-free with respect to $\mathbf{u}_0, \ldots, \mathbf{u}_j, t$.*
- *When the inputs $z_0, \ldots, z_{j-1}$ are specialized to $c_0, \ldots, c_{j-1}$, the circuit computes the truncated roots $\sum_{N \leq P} \rho_{\mathbf{c},i,N} t^N$ for each $i \leq j - 1$.*

Proof. We start by defining an auxiliary circuit family $\mathcal{C}'$. This family will be indexed by the parameters $n, j, d_0, \ldots, d_n, k, b_0, \ldots, b_{j-1}$. The input variables will be $\mathbf{u}_0, \ldots, \mathbf{u}_j, x_0, \ldots, x_{j-1}, z_0, \ldots, z_{j-1}, t$.

If $k \geq j$ or if $b_0 + \cdots + b_{j-1} > 2P$, then the corresponding circuit in $\mathcal{C}'$ is just the empty circuit. These conditions can be checked in time that is polynomial in the description of the index. We describe the circuit in the case where the indices satisfy $k < j$ and $b_0 + \cdots b_{j-1} \leq 2P$.

The circuit C' contains a copy of $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$ from Lemma 5.10. This computes $\widetilde{\mathbf{h}}_{\mathbf{z}}$ and its Jacobian determinant $\Delta_{\mathbf{z}}$. As usual, these gates are named $(1, v)$, where v is the name of the corresponding gate within $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$. The circuit then has b_0 copies of a sum gate, each of which computes $\widetilde{h}_{\mathbf{z},0}(\mathbf{x}) - x_0$. These are named $(2, 0, i)$ for $i \leq b_0$. Similarly, for each $e < j$, there are b_e copies of $\widetilde{h}_{\mathbf{z},t}(\mathbf{x}) - x_e$ computed by sum gates named $(2, e, i)$ for $i \leq b_e$. For each $e < j$, there is a product gate $(3, e)$ that multiplies these copies to compute $(h_{\mathbf{z},e}(\mathbf{x}) - x_e)^{b_e}$. Finally, there is a multiplication gate that has inputs $(3, e)$ for all e, the variable x_k, and the Jacobian determinant $\Delta_{\mathbf{z}}$ that was computed by the copy of $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$. The circuit therefore computes the polynomial

$$x_k(\widetilde{h}_{\mathbf{z},0}(\mathbf{x}) - x_0)^{b_0} \cdots (\widetilde{h}_{\mathbf{z},j-1}(\mathbf{x}) - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{z}}.$$

Uniformity of this circuit family follows from a simulation argument, since the gate names allow us to recover the gate names within the copy of $C^{\mathrm{scaled}}_{j,d_0,\ldots,d_{j-1}}$. No division gate involves division by a polynomial that depends on $\mathbf{x}$ or t. The size of the circuit C' is polynomial in P (note that U_j is itself polynomial in P). The degree in $\mathbf{x}$ and t of the polynomial computed by C' can likewise be bounded by $\mathrm{poly}(P)$.

Using the family $\mathcal{C}'$, we define another auxiliary circuit family $\mathcal{C}''$ with the same indices. The family $\mathcal{C}''$ is obtained by applying multivariate interpolation (Lemma 3.3) to $\mathcal{C}'$ to interpolate out the coefficients of $t, x_0, \ldots, x_{j-1}$. The statements at the end of the previous paragraph show that the assumptions required for the interpolation hold. To summarize the construction so far, we have a polylogtime-uniform family $\mathcal{C}''$ such that $C'' = C''_{n,j,d_0,\ldots,d_n,k,b_0,\ldots,b_{j-1}}$ has input variables $\mathbf{u}_0, \ldots, \mathbf{u}_{j-1}, z_0, \ldots, z_{j-1}$, and computes the coefficients of

$$x_k(\widetilde{h}_{\mathbf{z},0}(\mathbf{x}) - x_0)^{b_0} \cdots (\widetilde{h}_{\mathbf{z},j-1}(\mathbf{x}) - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{z}}$$

as a polynomial in $t, x_0, \ldots, x_{j-1}$. The size of C'' is bounded by $P^{\mathrm{poly}(n)}$, and the depth is a universal constant.

With this family in hand, we return to the construction of $C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$. Fix indices $n, j, d_0, \ldots, d_n$. We describe the circuit $C = C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$. For every vector $\mathbf{b} \in \mathbb{N}^j$ with $|\mathbf{b}| \leq 2P$ and for every $k < j$, the circuit C contains a copy of $C''_{n,j,d_0,\ldots,d_n,k,b_0,\ldots,b_{j-1}}$. We will call this subcircuit $C_{\mathbf{b},k}$ in the rest of this argument for brevity. The names of each gate in $C_{\mathbf{b},k}$ are of the form $(1, \mathbf{b}, k, v)$ where v is the name of the corresponding gate within the subcircuit. For each $\mathbf{b}, k$, and each $N \leq P$, the circuit C has a product gate that computes

$$(-1)^{|\mathbf{b}|} \cdot \mathrm{coeff}(x_i(\widetilde{h}_{\mathbf{z},0} - x_0)^{b_0} \cdots (\widetilde{h}_{\mathbf{z},j-1} - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{z}}, t^N x_0^{b_0} \cdots x_{j-1}^{b_{j-1}})$$

by taking as input a constant gate with constant -1 and the coefficient from the copy of $C_{\mathbf{b},k}$. These product gates are named $(2, \mathbf{b}, k, N)$. For each $1 \leq N \leq P$ and each $k < j$, the circuit C contains a sum gate that uses the above gates to compute

$$\rho_{\mathbf{z},k,N} \coloneqq \sum_{(b_0,\ldots,b_{j-1}),\ |\mathbf{b}| \leq 2N} (-1)^{|\mathbf{b}|} \cdot \mathrm{coeff}(x_k(\widetilde{h}_{\mathbf{z},0} - x_0)^{b_0} \cdots (\widetilde{h}_{\mathbf{z},j-1} - x_{j-1})^{b_{j-1}} \Delta_{\mathbf{z}}, t^N x_0^{b_0} \cdots x_{j-1}^{b_{j-1}}).$$

These gates are named $(3, N, k)$. The predecessors of the summation gates are numbered in lexicographic order based on $\mathbf{b}$. All that remains is to compute the constant terms $\rho_{\mathbf{z},k,0}$, which are simply $\mathbf{r}_{\mathbf{z}}$. For this, the circuit C contains a copy of $C^{\mathrm{init}}_{j,d_0,\ldots,d_{j-1}}$, whose gates are named $(4, v)$ where v is the name of the corresponding gate within the subcircuit. With all the coefficients $\rho_{\mathbf{z},i,N}$ computed as above, the circuit has product gates that compute $\rho_{\mathbf{z},k,N} t^N$ for each $N \leq P$ and $k \leq j-1$, and addition gates that add these to compute $\sum_{N \leq P} \rho_{\mathbf{z},k,N} t^N$ for each $k \leq j-1$. These gates have names of the form $(5, *)$.

We now bound the size of the circuit. The total number of vectors $\mathbf{b}$ with $|\mathbf{b}| \leq 2P$ is bounded by $P^{\mathrm{poly}(n)}$. The size of each copy $C_{\mathbf{b},k}$ is bounded by $P^{\mathrm{poly}(n)}$, and consequently the size of the whole circuit is bounded by $P^{\mathrm{poly}(n)}$. The fact that the depth is a universal constant is straightforward.

We now argue that the circuit family is uniform. Let T_C be the name of the machine that we will construct to decide the direct connection language of $\mathcal{C}^{\mathrm{root}}$. Let $T_{C'}$ and $T_{C''}$ be the machines that decide the direct connection languages of $\mathcal{C}'$ and $\mathcal{C}''$, respectively. Suppose $(n, j, d_0, \ldots, d_n, a, p, a')$ is an input to T_C. The bound P can be computed from the index in polynomial time. Suppose a has a name of the form $(1, \mathbf{b}, k, v)$. The key observation is that the machine T_C can simulate the machine $T_{C''}$ on inputs of the form $(n, j, d_0, \ldots, d_n, k, b_0, \ldots, b_{j-1}, v, p, a'')$, where a'' is either a gate type, or the name of a gate within $C_{\mathbf{b},k}$. This will allow T_C to use $T_{C'}$ to decide if $(n, j, d_0, \ldots, d_n, a, p, a')$ is a YES instance. As before, if v is an input or output gate of the subcircuit, then T_C uses the description in the construction above to check if the gate is wired correctly. The rest of the argument, and the argument when a is of the form $(2, *), (3, *), (4, *)$ or $(5, *)$ is the usual simulation argument that we have repeated before. □

With this, we can compute $\mathrm{Res}_n(\mathbf{F}_n)$ via a polylogtime-uniform family of constant-depth circuits.

Lemma 5.12. *There exists a polylogtime-uniform family of weakly division-free, constant-free circuits $\mathcal{C}^{\mathrm{Res}}$ indexed by parameters $n, d_0, \ldots, d_n$ with the following properties.*

- *The circuit $C^{\mathrm{Res}}_{n,d_0,\ldots,d_n}$ has size bounded by $P^{\mathrm{poly}(n)}$, where $P = \sum_{i=0}^{n} \prod_{j\neq i} d_j$, and depth bounded by a universal constant.*
- *The circuit $C^{\mathrm{Res}}_{n,d_0,\ldots,d_n}$ has $U := \sum_{i=0}^{n} \binom{n+d_i}{n}$ input gates, labeled by $\mathbf{u}_0 \ldots, \mathbf{u}_n$. The circuit has a single output gate.*
- *The circuit computes the multivariate resultant $\mathrm{Res}_n(\mathbf{F}_n)$ in the variables $\mathbf{u}_0, \ldots, \mathbf{u}_n$.*

Proof. We start by constructing an auxiliary family $\mathcal{C}'$ indexed by $n, d_0, \ldots, d_n$. Fix indices $n, d_0, \ldots, d_n$. The inputs to $C' = C'_{n,d_0,\ldots,d_n}$ will be variables $\mathbf{u}_0, \ldots, \mathbf{u}_n, t$.

We now describe C'. For each $j \in [n]$, the circuit C' has $d_0 \cdots d_{j-1}$ copies of the circuit $C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$ from Lemma 5.11 as sub-circuits. The names of the gates are $(1, j, i, v)$, where $i \leq d_0 \cdots d_{j-1}$ and v is the name of the corresponding gate within the subcircuit. In addition, the circuit C' has gates that compute each of the integers $1, \ldots, \sum_{i=1}^{n} d_i$.

The elements of $B_j = [d_0] \times \cdots \times [d_{j-1}]$ are tuples of integers of length j, and B_j itself has size $d_0 \cdots d_{j-1}$. Given $d_0, \ldots, d_n, j, n$, and i in binary, computing the i^{th} element of B_j in lexicographic order can be done in polynomial time. In C', the i^{th} copy of $C^{\mathrm{root}}_{n,j,d_0,\ldots,d_n}$ is evaluated at the i^{th} element of B_j using the integers $1, \ldots, \sum d_i$ computed above. By Lemma 5.11, each of the output gates of the copies computes approximations (in t) of the roots of the system $\mathbf{h}_j$.

The circuit C' also has copies of a subcircuit that computes the polynomials $h_{1,1}, \ldots, h_{n,n}$, as in the proof of Lemma 5.10. Specifically, it has one copy of $h_{j,j}$ for each element of B_j. To the input gates of the i^{th} copy of $h_{j,j}$, we wire the approximate roots constructed using the i^{th} element of B_j. Consequently, for each $\mathbf{c} \in B_j$, the circuit computes $h_{j,j}(\boldsymbol{\rho}'_{\mathbf{c}})$, where $\boldsymbol{\rho}'_{\mathbf{c}} = \boldsymbol{\rho}_{\mathbf{c}} \pmod{t^P}$. Finally, the circuit C' has gates that use these evaluations to implement the Poisson formula (Lemma 5.6).

To summarize, the circuit $C'_{n,d_0,\ldots,d_n}$ implements the Poisson formula for $\mathrm{Res}_n(\mathbf{H}_n)$, except it uses approximations of the roots of $\mathbf{h}_j$ in the ring $\mathbb{A}[\![t]\!]$. The order of approximation is $P = \sum_{i=0}^{n} \prod_{j\neq i} d_j$, and the output has degree $P^{\mathrm{poly}(n)}$. The fact that C' has size $P^{\mathrm{poly}(n)}$, constant depth, and the family $\mathcal{C}'$ is polylogtime-uniform follows by the same argument as in previous constructions. No division gate in C' involves division by a polynomial that depends on t. Further, the degree in t of the polynomial computed by C' is bounded by $P^{\mathrm{poly}(n)}$.

Applying Lemma 3.2 to the family $\mathcal{C}'$ with distinguished variable t results in a circuit family $\mathcal{C}''$ that computes the coefficients in t of the above approximation to $\mathrm{Res}_n(\mathbf{H}_n)$. The family $\mathcal{C}''$ is polylogtime-uniform and consists of circuits of size $P^{\mathrm{poly}(n)}$.

We now return to the construction of $C^{\mathrm{Res}}_{n,d_0,\ldots,d_n}$. This circuit has a copy of $C''_{n,d_0,\ldots,d_n}$. It has one additional summation gate that sums the outputs of $C''_{n,d_0,\ldots,d_n}$ corresponding to the coefficients of t^i for $i \leq P$. The computed polynomial is the same as the one obtained by truncating the approximate computation of the resultant and evaluating at $t = 1$. By the Poisson formula, the degree bounds on the resultant, and the construction of the homotopy, this polynomial is exactly $\mathrm{Res}_n(\mathbf{F}_n)$. The claims on the size, depth, and uniformity are again straightforward. □

In Sections 4.2 and 4.3, we saw examples of how the resultant can be used to solve problems beyond satisfiability of square homogeneous systems. In these applications, we often have to compute the resultant of a set of polynomials whose coefficients are themselves polynomials. If these coefficients come from the ring $\mathbb{Q}[w_1, \ldots, w_k]$, then the resultant is itself a polynomial in $\mathbb{Q}[w_1, \ldots, w_k]$, and can be computed by simply substituting these polynomial coefficients in the circuit for $\mathrm{Res}_n(\mathbf{F}_n)$ designed above.

If the coefficients are polynomials in $w_1, \ldots, w_k$ of degree at most D, then the resulting computation can also be seen as taking $U \cdot \binom{k+D}{k}$ inputs, one corresponding to each monomial in the $\mathbf{w}$ variables in each coefficient of $\mathbf{F}_n$, and having at most $(Dnd^n)^{O(k)}$ outputs, where $d \coloneqq \max_i d_i$. The outputs correspond to the coefficients of the $\mathbf{w}$ variables in the computed resultant. The following corollary states that this computation can be carried out by polylogtime-uniform circuits of constant depth. It is an easy corollary of the above resultant computation and the uniformity of interpolation.

Corollary 5.13. *There exists a polylogtime-uniform family of weakly division-free, constant-free circuits $\mathcal{C}^{\text{composed}}$ indexed by parameters $n, d_0, \ldots, d_n, k, D$ with the following properties.*

- *The circuit $C^{\text{composed}}_{n,d_0,\ldots,d_n,k,D}$ has size bounded by $(PD)^{\text{poly}(n,k)}$, where $P = \sum_{i=0}^{n} \prod_{j \neq i} d_j$, and depth bounded by a universal constant.*
- *The circuit has $U \cdot \binom{k+D}{k}$ input variables, which we denote by $u_{i,\boldsymbol{\alpha},\boldsymbol{\beta}}$ (where $\boldsymbol{\beta} \in \mathbb{N}^k$ satisfies $|\boldsymbol{\beta}| \leq D$). The circuit has $\binom{DD'+k}{k}$ outputs, where D' is the degree of $\operatorname{Res}_n(\mathbf{F}_n)$.*
- *When evaluated at $\gamma_{i,\boldsymbol{\alpha},\boldsymbol{\beta}}$, the output gates compute the coefficients of the monomials in $\mathbf{w}$ of the resultant of the polynomials*

$$H_i = \sum_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} \gamma_{i,\boldsymbol{\alpha},\boldsymbol{\beta}} \mathbf{x}^{\boldsymbol{\alpha}} \mathbf{w}^{\boldsymbol{\beta}}$$

with respect to the variables $\mathbf{x}$.

Proof. There exist polylogtime-uniform constant-depth circuits that compute the polynomials $\sum_{\boldsymbol{\beta}} u_{i,\boldsymbol{\alpha},\boldsymbol{\beta}} \mathbf{w}^{\boldsymbol{\beta}}$ for each i and $\boldsymbol{\alpha}$. We then compose these with the circuit for $\operatorname{Res}_n(\mathbf{F}_n)$ obtained from Lemma 5.12. The resulting circuit is polylogtime-uniform and computes a polynomial in the variables $\mathbf{w}$ and $u_{i,\boldsymbol{\alpha},\boldsymbol{\beta}}$. The degree in the $\mathbf{w}$ variables is at most DD'. We use Lemma 3.3 to interpolate out the coefficients of monomials in $\mathbf{w}$. This resulting circuit satisfies the depth, size, and uniformity requirements that we claim. □

6 Deciding the Nullstellensatz in the counting hierarchy

In this section, we prove that Hilbert's Nullstellensatz can be decided in the counting hierarchy. To do this, we prove a general transfer theorem that translates uniform constructions of arithmetic circuits into algorithms that can be run on Turing machines.

To every family of polynomials $\mathcal{F} = (f_{\mathbf{n}})_{\mathbf{n}}$ with rational coefficients, there is an associated family of boolean functions $\hat{\mathcal{F}} = (\hat{f}_{\mathbf{n},h})_{\mathbf{n},h}$, where $\hat{f}_{\mathbf{n},h}$ corresponds to evaluation of $f_{\mathbf{n}}$ on tuples of rational numbers represented by pairs of integers of height at most h. Our goal will be to show that if the family of polynomials $\mathcal{F}$ can be computed by polylogtime-uniform arithmetic circuits of constant depth and exponential size, then this associated family of boolean functions can be computed in the counting hierarchy CH. Once we have this in hand, an immediate corollary of Lemma 5.12 will be an $\mathsf{FP}^{\mathsf{CH}}$ algorithm to evaluate the multivariate resultant. Applying the reductions of Propositions 4.4 and 4.12 will yield CH and $\mathsf{FP}^{\mathsf{CH}}$ algorithms for the decision and counting versions of Hilbert's Nullstellensatz, respectively.

6.1 From uniform arithmetic circuits to the counting hierarchy

Our proof of the transfer theorem proceeds in two steps. We first use the polylogtime-uniform arithmetic circuits for the family $\mathcal{F}$ to construct polylogtime-uniform threshold circuits of similar

size and depth for the boolean functions $\hat{f}_{\mathbf{n},h}$. The fact that arithmetic circuits can be simulated efficiently by threshold circuits is a straightforward consequence of the fact that iterated addition and iterated multiplication are in logtime-uniform TC^0 (items 1 and 2 of Theorem 2.12, respectively). The uniformity of the resulting threshold circuits is a consequence of the logtime-uniformity of iterated addition and multiplication combined with the assumed uniformity of the arithmetic circuits that compute $\mathcal{F}$.

We then show that this family of threshold circuits for $\hat{f}_{\mathbf{n},h}$ can be evaluated in CH. Even though these threshold circuits are exponentially large, the fact that they are uniform means that a polynomial-time Turing machine can use nested majority quantifiers to simulate them. The number of majority quantifiers used in this simulation corresponds to the depth of the threshold circuit that computes $\hat{f}_{\mathbf{n},h}$. Since we build threshold circuits of constant depth for $\hat{f}_{\mathbf{n},h}$, the resulting evaluation procedure uses a bounded number of majority quantifiers, and hence lies in the counting hierarchy.

Neither of these ideas are fundamentally new to our work. Agrawal, Allender, and Datta [AAD00] observed that arithmetic circuits can be transformed into threshold circuits that evaluate the corresponding polynomial on $\{0,1\}$-valued inputs, and that this transformation preserves the size and depth of the circuit. Allender, Koucký, Ronneburger, Roy, and Vinay [AKRRV01] proved a result similar to ours, showing that exponentially-large constant-depth arithmetic circuits can be converted to CH algorithms. This second result uses threshold circuits as an intermediate representation in the same manner as our work.

Although these ideas are present in prior work, we are not aware of any reference that starts with a uniform family of arithmetic circuits for a polynomial family $\mathcal{F}$ and concludes that the corresponding boolean functions can be computed in CH. Because of this, we provide complete details for the results of this section.

We start by converting arithmetic circuits into threshold circuits by implementing each arithmetic gate as a TC^0 sub-circuit using Theorem 2.12. Because the circuits provided by Theorem 2.12 are uniform, this procedure preserves the uniformity present in the family of arithmetic circuits. The families of boolean functions we construct throughout this subsection treat their inputs as sequences of rational numbers, represented as pairs of integers. To make these functions total rather than partial, we adopt the convention that if any such pair of integers is of the form $(a, 0)$, the boolean function evaluates to zero. In what follows, we tacitly assume that inputs are well-formed, since this can be decided in polynomial time as a preprocessing step.

Lemma 6.1. *Let $\mathcal{C} = (C_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of weakly division-free, constant-free arithmetic circuits over $\mathbb{Q}$ of size $s_{\mathbf{n}}$ and depth $\Delta \in \mathbb{N}$. Let $f_{\mathbf{n}} \in \mathbb{Q}[x_1, \ldots, x_{m_{\mathbf{n}}}]$ be the polynomial computed by $C_{\mathbf{n}}$. There exists a family of boolean functions $\hat{f}_{\mathbf{n},h}$ indexed by $\mathbf{n}$ and a natural number h with the following properties.*

- *For each $\mathbf{n}$ and h, the function $\hat{f}_{\mathbf{n},h}$ has $2m_{\mathbf{n}}(h+1)$ inputs, which are interpreted as $m_{\mathbf{n}}$ rational numbers represented by a pair of integers of height h. Further, the function has $2h \cdot (s_{\mathbf{n}} \log s_{\mathbf{n}})^{\Delta} + 2$ outputs, which are interpreted as a single rational number represented by a pair of integers of height $h \cdot (s_{\mathbf{n}} \log s_{\mathbf{n}})^{\Delta}$.*
- *If $(a_1, b_1), \ldots, (a_{m_{\mathbf{n}}}, b_{m_{\mathbf{n}}})$ are representations of rational numbers of height at most h, then $\hat{f}_{\mathbf{n},h}((a_1, b_1), \ldots, (a_{m_{\mathbf{n}}}, b_{m_{\mathbf{n}}}))$ is a representation of $f_{\mathbf{n}}(a_1/b_1, \ldots, a_{m_{\mathbf{n}}}/b_{m_{\mathbf{n}}})$.*
- *There exists a polylogtime-uniform family $\mathcal{D} = (D_{\mathbf{n},h})_{\mathbf{n},h}$ of threshold circuits of depth $O(\Delta)$ and size $\mathrm{poly}\left(h, (s_{\mathbf{n}} \log s_{\mathbf{n}})^{\Delta}\right)$ that compute the functions $\hat{f}_{\mathbf{n},h}$.*

Proof. We will construct circuits for $\hat{f}_{\mathbf{n},h}$ from the circuit $C_{\mathbf{n}}$ by replacing each arithmetic gate with a threshold circuit that implements iterated addition and iterated multiplication of rational numbers,

using items 1 and 2 of Theorem 2.12. To do this, we first need to bound the heights of the integers used to represent intermediate values in $C_{\mathbf{n}}$, as this will determine the size of the threshold circuits we need to correctly simulate the operations done in $C_{\mathbf{n}}$. For each gate v in the circuit $C_{\mathbf{n}}$, we will compute a rational number a_v/b_v that represents the value of the gate on the given input. We set $C = C_{\mathbf{n}}$ and $s = s_{\mathbf{n}}$ in the next part of this discussion for brevity.

For a gate v, the *depth* of v is the length of the longest path starting from v and ending at either an input gate or a constant gate. For example, input gates and constant gates have depth 0, while a gate that only has constant gates as input has depth 1. We claim that if v has depth δ in C, then whenever the inputs to C are represented by pairs of integers of height at most h, the values of a_v and b_v can be represented by integers of height bounded by $h \cdot (s \log s)^{\delta}$. We prove this by induction on δ. When $\delta = 0$, the gate v is either an input to the circuit or one of the constants $0, +1, -1$. If v is a constant, then clearly the heights of a_v and b_v are at most 1. If v is an input, then by assumption, the numerator a_v and denominator b_v both have height at most h.

When $\delta \geq 1$, we proceed by case analysis depending on the operation labeling the gate v.

- Suppose $v = v_1 + \cdots + v_k$ is an addition gate. Because the circuit C has size bounded by s, we know $k \leq s$. Each v_i is represented by a quotient a_i/b_i where, by induction, the integers a_i and b_i have height at most $h \cdot (s \log s)^{\delta-1}$. We represent the value of v as a/b, where

$$a \coloneqq \left(\frac{a_1}{b_1} + \cdots + \frac{a_k}{b_k}\right) b_1 \cdots b_k = a_1 b_2 \cdots b_k + b_1 a_2 b_3 \cdots b_k + \ \cdots \ + b_1 \cdots b_{k-1} a_k$$
$$b \coloneqq b_1 \cdots b_k.$$

  Every product appearing in the definitions of a and b above is a product of k numbers, each of which have height $h \cdot (s \log s)^{\delta-1}$. This means that each such product—and in particular, the integer b—has height $kh \cdot (s \log s)^{\delta-1}$. The integer a is a sum of k such terms, so has height $k \log(k) h \cdot (s \log s)^{\delta-1}$. The fact that $k \leq s$ implies that a and b have height at most $h \cdot (s \log s)^{\delta}$, as claimed.

- Suppose $v = v_1 \times \cdots \times v_k$ is a product gate. As in the previous case, we know $k \leq s$. Each v_i is represented by a quotient a_i/b_i where, by induction, the integers a_i and b_i have height at most $h \cdot (s \log s)^{\delta-1}$. We represent the value of v as a/b, where

$$a \coloneqq a_1 \cdots a_k$$
$$b \coloneqq b_1 \cdots b_k.$$

  The same analysis as in the case where v is an addition gate shows that a and b have height at most $h \cdot (s \log s)^{\delta}$.

- Finally, suppose $v = v_1/v_2$ is a division gate. The values of v_1 and v_2 are represented by the quotients a_1/b_1 and a_2/b_2, respectively. By induction, each a_i and b_i has height at most $h \cdot (s \log s)^{\delta-1}$. We represent v as the quotient a/b, where

$$a \coloneqq a_1 b_2$$
$$b \coloneqq a_2 b_1.$$

  It is clear that both a and b have height at most $2h \cdot (s \log s)^{\delta-1} \leq h \cdot (s \log s)^{\delta}$, as claimed.

To construct the circuit for the boolean function $\hat{f}_{\mathbf{n},h}$, we replace each of the arithmetic gates in C with a threshold circuit that implements the corresponding operation, using the TC^0 circuits

provided by Theorem 2.12. We also replace each of the constant gates by a set of 1 and 0 gates that encode representations of the respective constant. As the analysis above shows, it suffices to use threshold circuits that implement iterated addition and iterated multiplication of at most s integers of height $h \cdot (s \log s)^{\Delta}$, since this is the maximum height required to represent the value of any gate in the circuit when evaluated on an input of height at most h. These operations can be performed with threshold circuits of constant depth and size $h^{O(1)}(s \log s)^{O(\Delta)}$. The fact that the arithmetic circuit C is weakly division-free ensures that as long as all the inputs are well-formed, no division by zero takes place when evaluating the circuit in the above manner.

Formally, the threshold circuits provided by Theorem 2.12 perform iterated addition or multiplication of $h \cdot (s \log s)^{\Delta}$ integers of height $h \cdot (s \log s)^{\Delta}$, even though gates in C have fan-in bounded by s. To deal with this, we hardwire the superfluous inputs of these threshold subcircuits to the constant 0 for iterated addition and the constant 1 for iterated multiplication. Similarly, these threshold circuits will have more than $h \cdot (s \log s)^{\Delta}$ bits of output. We ignore these extra bits of output, since we know they will not affect the result of our simulation.

Since the circuit C has size s and depth Δ, it is clear that this procedure results in a threshold circuit of size $h^{O(1)}(s \log s)^{O(\Delta)}$ and depth $O(\Delta)$ that correctly computes a binary representation of the output of the circuit C on inputs of height h. We denote the resulting threshold circuit by $D_{\mathbf{n},h}$ and define the boolean function $\hat{f}_{\mathbf{n},h}$ to be the function computed by $D_{\mathbf{n},h}$.

It remains to bound the uniformity of the circuit family $\mathcal{D} = (D_{\mathbf{n},h})_{\mathbf{n},h}$. The names of gates in $D_{\mathbf{n},h}$ that are part of threshold circuits simulating arithmetic will be a tuple $(1, v, u)$, where v is the name of the gate in $C_{\mathbf{n}}$ that is being simulated and u is the name of a gate within the threshold subcircuit that simulates the arithmetic of v. The other gates in $D_{\mathbf{n},h}$, namely those that compute encodings of the integers 1 and 0 to be used in the extra inputs to the arithmetic subcircuits, have names of the form $(2, *)$. Let T_C, T_+, and $T_\times$ be Turing machines that decide the direct connection languages of $\mathcal{C}$, threshold circuits for iterated addition, and threshold circuits for iterated multiplication, respectively. Denote by T_D the Turing machine that we will design to decide the direct connection language of $\mathcal{D}$.

Let $(\mathbf{n}, h, a, p, b)$ be an input to T_D. Suppose $a = (1, v_a, u_a)$. If $p = \epsilon$, then T_D must accept if b represents the type of a. To decide this, the machine T_D first computes the type of the gate v_a using T_C. If v_a is a summation gate, then T_D simulates T_+ to compute the type of u_a and accepts if b matches this type. If v_a is a product gate, then T_D simulates either T_+ or $T_\times$, as appropriate, to compute the type of u_a and accept or reject accordingly. If v_a is a division gate, then T_D simulates $T_\times$ to compute the type of u_a. If v_a is an input gate of the subcircuit but a is not an input gate of $D_{\mathbf{n},h}$, then we adopt the (arbitrary) convention that the type of a is OR.

If $p \neq \epsilon$, then the machine T_D branches based on b.

- T_D first checks if b is of the form $(1, v_b, u_b)$. In this case, it branches further as follows.
  - Suppose $v_a = v_b$. In this case, a and b are part of the same subcircuit implementing arithmetic. The machine computes the type of the gate v_a as in the above paragraph, and uses the corresponding Turing machine to check if u_b is a predecessor of u_a. If so, it accepts, and if not then it rejects.
  - Suppose $v_a \neq v_b$. In this case, the machine checks if u_a is an input gate in the threshold subcircuit, and rejects if it is not. If it is, then T_D computes integers p_1 and p_2 such that u_a is the $p_1{}^{\text{th}}$ bit of the $p_2{}^{\text{th}}$ rational number input to the threshold subcircuit. It then computes the arity of v_a and rejects if the arity is less than p_2. If the arity is more than p_2, the machine uses T_C to check if v_b is the $p_2{}^{\text{th}}$ input to v_a in $C_{\mathbf{n}}$. It also checks if u_b is

the $p_1{}^{\text{th}}$ output bit of its corresponding threshold subcircuit. If all of these checks pass, it accepts the string. If any of them fail, it rejects.

- Suppose b has name $(2, *)$. In this case, as above, the machine T_D checks that a is an input bit. It then computes integers p_1 and p_2 such that u_a is the $p_1{}^{\text{th}}$ bit of the $p_2{}^{\text{th}}$ integer input to the threshold circuit. The machine T_D verifies that the arity of v_a is at most p_2. Finally, T_D computes the type of a and, based on whether a is a summation or multiplication gate, checks that b is the correct bit in the encoding of the 0 or 1 that is used for the extra inputs. If all these checks pass, T_D accepts its input.

The case when $a = (2, *)$ can be handled in a similar manner. All computations take time polylogarithmic in the size of the final circuit and polynomial in the binary representation of the index. This shows that the circuit family $\mathcal{D}$ is polylogtime-uniform. □

While Lemma 6.1 is stated for single output circuit families, it can be easily extended to circuit families with multiple outputs. In this case, the boolean function $f_{\mathbf{n},h}$ computes a tuple of binary representations of natural numbers, one for each output gate of $C_{\mathbf{n}}$. The proof itself requires no changes.

Next, we show that uniform threshold circuits of constant depth and exponential size can be evaluated in the counting hierarchy. Of course, the number of output gates in such a circuit may itself be exponential, so it may not be possible to write down the complete output in polynomial time. We instead show that given a multi-index $\mathbf{n}$, an input to the $\mathbf{n}^{\text{th}}$ circuit in the family, and an index i, we can compute the i^{th} bit of the corresponding output in CH, a problem we formalize below.

Definition 6.2. Let $\mathcal{F} \coloneqq \left(f_{\mathbf{n}} : \{0,1\}^{m'_{\mathbf{n}}} \to \{0,1\}^{m_{\mathbf{n}}}\right)_{\mathbf{n}}$ be a multi-indexed family of boolean functions. The *language $L_{\mathcal{F}}$ corresponding to $\mathcal{F}$* is defined as

$$L_{\mathcal{F}} \coloneqq \{(i, b, x, \mathbf{n}) \mid f_{\mathbf{n}}(x)_i = b\}.$$

If $\mathcal{D}$ is a family of circuits, then we abuse notation and use $L_{\mathcal{D}}$ to denote the language corresponding to the family of functions computed by $\mathcal{D}$. ◊

Next, we show that if $\mathcal{D}$ is a polylogtime-uniform family of exponentially-large threshold circuits, then the language $L_{\mathcal{D}}$ corresponding to $\mathcal{D}$ lies in the counting hierarchy.

Lemma 6.3 (see, e.g., [ABKPM09, Proof of Theorem 4.1])**.** *Let $\mathcal{D} = (D_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of threshold circuits of size $s_{\mathbf{n}}$ and depth Δ, where $D_{\mathbf{n}}$ has $m_{\mathbf{n}}$ inputs. Let $L_{\mathcal{D}}$ be the language corresponding to the circuit family $\mathcal{D}$ as defined in Definition 6.2. Suppose $s_{\mathbf{n}} \leq \exp(\text{poly}(m_{\mathbf{n}}))$. Then $L_{\mathcal{D}} \in \mathsf{CH}$.*

Proof. For an integer $\delta \in \mathbb{N}$, consider the auxiliary language

$$L_\delta \coloneqq \{(g, b, x, \mathbf{n}) \mid g \text{ is a gate of depth at most } \delta \text{ in the circuit } D_{\mathbf{n}} \text{ and evaluates to } b \text{ on input } x\},$$

where the gate name g and index $\mathbf{n}$ into the circuit family are provided in binary. We will prove by induction on δ that $L_\delta \in \mathsf{CH}$.

As a first step, we show that given $(g, b, x, \mathbf{n})$ as input, we can check if x has length $m_{\mathbf{n}}$ in time polynomial in the length of the input. Recall that, by the definition of uniformity, there is a polynomially-bounded increasing function $T : \mathbb{N} \to \mathbb{N}$ and a Turing machine M such that given input $\mathbf{n}$, the machine M runs in time at most $T(\log(s_{\mathbf{n}}) + N)$ and computes $m_{\mathbf{n}}$, where $N = \sum_i \log n_i$.

To decide if x has length $m_{\mathbf{n}}$, we are only allowed time polynomial in $|g| + |x| + N$, rather than $\log(s_{\mathbf{n}}) + N$. Let Q be a polynomially-bounded increasing function such that $\log(s_{\mathbf{n}}) \leq Q(m_{\mathbf{n}})$. To check if x has the correct length, we simulate M on input $\mathbf{n}$ for at most $T(Q(|x|) + N)$ steps. If this simulation terminates, we obtain $m_{\mathbf{n}}$ and can check if $|x| = m_{\mathbf{n}}$. If this simulation does not terminate, then we know that x is too short to be a valid input to $C_{\mathbf{n}}$. When x has the correct length, the fact that $\log(s_{\mathbf{n}}) \leq Q(m_{\mathbf{n}})$ also implies that the name of every gate in $C_{\mathbf{n}}$ has length bounded by a polynomial function of the length of the input $(g, b, x, \mathbf{n})$. In the rest of this proof, all our machines will perform the above check first and reject if it fails.

We now consider the language L_0. We receive $(g, b, x, \mathbf{n})$ as input and must decide if g is a gate labeled by a constant or if g is an input gate of $D_{\mathbf{n}}$ and, if so, whether g evaluates to b on input x. By the assumed size and uniformity of the circuit family $\mathcal{D}$, we can decide if g is an input gate in polynomial time (in the length of $(g, b, x, \mathbf{n})$) and, if so, determine which input bit x_i labels g. In this case we accept $(g, b, x, \mathbf{n})$ if and only if $b = x_i$, where x_i is the input bit that labels g. If g is not an input gate, we can check if it has type 0, $+1$, or -1, and accept if b matches the type. We reject all other inputs. This shows that $L_0 \in \mathsf{P} \subseteq \mathsf{CH}$.

When $\delta \geq 1$, we will show that the language L_δ can be decided in PP with oracle access to $L_{\delta-1}$. As $L_{\delta-1} \in \mathsf{CH}$ by induction, we can conclude that $L_\delta \in \mathsf{CH}$. Let $(g, b, x, \mathbf{n})$ be the input to L_δ. By the assumed size and uniformity of $\mathcal{D}$, we can determine the type of the gate g in time polynomial in the length of $(g, b, x, \mathbf{n})$. Without loss of generality, we may assume that $\mathcal{D}$ consists only of majority and negation gates, since the case of input and constant gates can be handled as above.

- Suppose g is a negation gate. We first nondeterministically guess a gate h and verify, using the uniformity of $\mathcal{D}$, whether h is the child of g. The fact that the names of all gates are bounded by a polynomial function of the length of the input $(g, b, x, \mathbf{n})$ allows us to perform this step. If h is not the child of g, then we nondeterministically branch once more, accepting in one path and rejecting in the other, so that the branches corresponding to non-children of g contribute the same number of accepting and rejecting paths. If h is the child of g, then we use oracle access to $L_{\delta-1}$ to check if $(h, 1 - b, x, \mathbf{n}) \in L_{\delta-1}$, accepting $(g, b, x, \mathbf{n})$ if and only if $(h, 1 - b, x, \mathbf{n}) \in L_{\delta-1}$. The majority of computation paths are accepting exactly when g evaluates to b on input x, so we can decide if $(g, b, x, \mathbf{n}) \in L_\delta$ using a $\mathsf{PP}^{L_{\delta-1}}$ algorithm.

- Suppose g is a majority gate. As in the previous case, we nondeterministically guess a gate h and verify if h is a child of g, again doing this using the uniformity of $\mathcal{D}$. If h is not a child of g, we branch into one accepting and one rejecting path so the non-children of g contribute an equal number of accepting and rejecting paths. If h is a child of g, we use the $L_{\delta-1}$ oracle to decide if h evaluates to b on input x and accept if and only if this is the case. The majority of computation paths are accepting exactly when the majority of the children of g evaluate to b on input x, so we can decide if $(g, b, x, \mathbf{n}) \in L_\delta$ using a $\mathsf{PP}^{L_{\delta-1}}$ algorithm.

In both cases, we can decide if $(g, b, x, \mathbf{n}) \in L_\delta$ in $\mathsf{PP}^{L_{\delta-1}}$ as claimed. As a consequence, we obtain $L_\Delta \in \mathsf{CH}$.

We now consider $L_{\mathcal{D}}$. Given an input $(i, b, x, \mathbf{n})$, the uniformity of $\mathcal{D}$ also allows us to compute the name of the output gate corresponding to the index i. We can then decide if the string is a YES instance using an oracle to L_Δ. This proves $L_{\mathcal{D}} \in \mathsf{CH}$. □

By combining the preceding lemmas, we conclude that if a family of polynomials $\mathcal{F}$ can be computed by a uniform family of arithmetic circuits of exponential size and constant depth, and if the corresponding family of boolean functions is $\hat{\mathcal{F}}$, then $L_{\hat{\mathcal{F}}}$ is in CH.

Corollary 6.4. *Let $\mathcal{C} = (C_{\mathbf{n}})_{\mathbf{n}}$ be a polylogtime-uniform family of weakly division-free, constant-free, constant-depth arithmetic circuits of size $s_{\mathbf{n}}$, where $C_{\mathbf{n}}$ has $m_{\mathbf{n}}$ inputs. Let $\mathcal{D}$ denote the circuit family constructed in Lemma 6.1 that computes boolean functions corresponding to the polynomials computed by $\mathcal{C}$. Suppose that $s_{\mathbf{n}} \leq \exp(\mathrm{poly}(m_{\mathbf{n}}))$. Then $L_{\mathcal{D}} \in \mathsf{CH}$.*

Proof. Combining the assumptions on the size of the circuits in $\mathcal{C}$ with the bounds on the size of $\mathcal{D}$ guaranteed by Lemma 6.1, we see that $\mathcal{D}$ satisfies the assumptions of Lemma 6.3. The result follows by an application of Lemma 6.3 to $\mathcal{D}$. □

We need one final simulation result. Suppose we have a family of functions $(f_n)_n$ that map n bits to 2^n bits. Suppose we also have a polylogtime-uniform family $\mathcal{D} = (D_n)_n$ of polynomial-sized TC^0 circuits where D_n takes n bits as input. In this setting, we can define a family of functions obtained by composing D_{2^n} with f_n for each n. The following lemma states that the language corresponding to this composed function family can be decided in $\mathsf{CH}^{L_{\mathcal{F}}}$.

Lemma 6.5. *Let $\mathcal{D} = (D_n)_n$ be a polylogtime-uniform family of polynomial-size threshold circuits of depth Δ, where D_n has n inputs. Let $\mathcal{F} = \left(f_{\mathbf{n}} : \{0,1\}^{m'_{\mathbf{n}}} \to \{0,1\}^{m_{\mathbf{n}}}\right)_{\mathbf{n}}$ be a family of boolean functions. Suppose $m_{\mathbf{n}} \leq \exp(\mathrm{poly}(m'_{\mathbf{n}}))$ and that a binary representation of $m_{\mathbf{n}}$ can be computed in polynomial time given a unary representation of $m'_{\mathbf{n}}$ and a binary representation of $\mathbf{n}$. Let $\mathcal{D} \circ \mathcal{F} = (D_{m_{\mathbf{n}}} \circ f_{\mathbf{n}})_{\mathbf{n}}$ denote the family of composed functions. Then $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}^{L_{\mathcal{F}}}$. In particular, if $L_{\mathcal{F}} \in \mathsf{CH}$, then $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}$.*

Proof. We proceed as in the proof of Lemma 6.3. For $\delta \in \mathbb{N}$, consider the auxiliary language

$$L_\delta \coloneqq \{(g, b, x, \mathbf{n}) \mid g \text{ is a gate of depth at most } \delta \text{ in } D_{m_{\mathbf{n}}} \text{ and evaluates to } b \text{ on input } f_{\mathbf{n}}(x)\}$$

By induction on δ, we will show that $L_\delta \in \mathsf{CH}^{L_{\mathcal{F}}}$.

As a first step, we query $L_{\mathcal{F}}$ on the inputs $(1, 1, x, \mathbf{n})$ and $(1, 0, x, \mathbf{n})$. The length of x is equal to $m'_{\mathbf{n}}$ if and only if exactly one of these two strings is in $L_{\mathcal{F}}$, so these queries allow us to ensure that x has the correct length to be a valid input to $f_{\mathbf{n}}$. Whenever x has the correct length, we have $m_{\mathbf{n}} \leq \exp(\mathrm{poly}(|x|))$ by assumption. In the rest of this argument, all our machines will perform the above check.

We now consider the case when $\delta = 0$. We receive $(g, b, x, \mathbf{n})$ as input and must decide if g is an input gate of $D_{m_{\mathbf{n}}}$ and, if so, whether g evaluates to b on input $f_{\mathbf{n}}(x)$. Because the circuit family $\mathcal{D}$ is polylogtime-uniform, and by the observation at the end of the previous paragraph, we can decide if g is an input gate in time polynomial in the length of the input $(g, b, x, \mathbf{n})$. If g is the p^{th} input gate, then we use an oracle call to $L_{\mathcal{F}}$ to decide if $(p, b, x, \mathbf{n}) \in L_{\mathcal{F}}$ and accept if and only if this oracle call returns YES. If g is a gate of type 0 or 1, then we accept or reject depending on b. This shows that $L_\delta \in \mathsf{P}^{L_{\mathcal{F}}} \subseteq \mathsf{CH}^{L_{\mathcal{F}}}$.

For $\delta \geq 1$, the same argument as in the proof of Lemma 6.3 shows that L_δ can be decided in PP with oracle access to $L_{\delta-1}$. By induction, we have $L_{\delta-1} \in \mathsf{CH}^{L_{\mathcal{F}}}$, so this implies that $L_\delta \in \mathsf{PP}^{\mathsf{CH}^{L_{\mathcal{F}}}} = \mathsf{CH}^{L_{\mathcal{F}}}$ as desired.

In particular, by taking $\delta = \Delta$ and using uniformity of $\mathcal{D}$ as in the proof of Lemma 6.3, we obtain $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}^{L_{\mathcal{F}}}$. If in addition $L_{\mathcal{F}} \in \mathsf{CH}$, then we have $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}^{\mathsf{CH}} = \mathsf{CH}$ as claimed. □

A frequent use case of Lemma 6.5 will be to compose two uniform families of threshold circuits, where the inner family is of exponential size and the outer family is of polynomial size. By Lemma 6.3, the output gates of the inner family can be evaluated in CH. Combining this with Lemma 6.5 above shows that the composition can also be evaluated in CH.

We will also encounter the following extension of the above situation. We have a family of circuits of exponential size, with multiple outputs. These outputs will be naturally grouped together, and we want to compose each group of outputs with a polynomial sized circuit. For example, the first circuit family computes the coefficients of a polynomial with coefficients in $\mathbb{Z}$, and we want to reduce each coefficient modulo a prime p. In this setting, we can derive the same conclusion as Lemma 6.5, under the same assumptions. Essentially the same proof as that of Lemma 6.5 applies in this setting, therefore we omit the proof.

Lemma 6.6. *Let $\mathcal{D} = (D_n)_n$ be a polylogtime-uniform family of polynomial-size threshold circuits of depth Δ, where D_n has n inputs. Let $\mathcal{F} = \left(f_{\mathbf{n}} : \{0,1\}^{m'_{\mathbf{n}}} \to \{0,1\}^{m^{(1)}_{\mathbf{n}} \times m^{(2)}_{\mathbf{n}}}\right)_{\mathbf{n}}$ be a family of boolean functions. Suppose $m_{\mathbf{n}} \leq \exp(\mathrm{poly}(m'_{\mathbf{n}}))$ and that binary representations of $m^{(1)}_{\mathbf{n}}$ and $m^{(2)}_{\mathbf{n}}$ can be computed in polynomial time from a unary representation of $m'_{\mathbf{n}}$ and a binary representation of $\mathbf{n}$. Let $\mathcal{D} \circ \mathcal{F}$ denote the family of functions obtained by composing each of the $m^{(2)}_{\mathbf{n}}$ sets of outputs of $f_{\mathbf{n}}$ with the function computed by $D_{m^{(1)}_{\mathbf{n}}}$. Then $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}^{L_{\mathcal{F}}}$. In particular, if $L_{\mathcal{F}} \in \mathsf{CH}$, then $L_{\mathcal{D}\circ\mathcal{F}} \in \mathsf{CH}$.*

6.2 Computing the resultant and deciding the Nullstellensatz

Applying Corollary 6.4 to the uniform circuit family for the multivariate resultant obtained in Lemma 5.12, we conclude that the multivariate resultant over $\mathbb{Z}$ can be evaluated in the counting hierarchy. Using this algorithm for the resultant over $\mathbb{Z}$ as a starting point, we can likewise conclude that the resultant over other domains, such as $\mathbb{F}_p$, can be computed in the counting hierarchy. This uses the fact that the resultant over $\mathbb{Z}$ has integer coefficients and that its image modulo p is precisely the resultant over $\mathbb{F}_p$ (see Definition 4.1). Because of this, we can compute the resultant over $\mathbb{F}_p$ by first lifting the input to polynomials with integer coefficients, then computing the resultant over the integers, and finally reducing the result modulo p. Similar ideas allow us to compute resultants over $\mathbb{F}_{p^a}$ and $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$.

Recall that in Section 2.3 we discussed various integral domains and how elements of these domains can be represented in binary. If R is any such domain, and if $f \in R[x_1, \ldots, x_n]$ is a polynomial, then we can define boolean functions $\hat{f}_h$ corresponding to evaluation of f on inputs of height at most h, analogous to the previous section where we did this for $R = \mathbb{Z}$ and $R = \mathbb{Q}$.

Theorem 6.7. *Let R be one of the rings $\mathbb{Z}, \mathbb{Z}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Let $\mathcal{F}_R$ be the family of boolean functions corresponding to the resultant over R. Then $L_{\mathcal{F}_R} \in \mathsf{CH}$.*

Proof. We proceed by case analysis depending on the choice of the ring R.

- $R = \mathbb{Z}$: Apply Corollary 6.4 to the circuits of Lemma 5.12. The statement of Lemma 5.12 provides bounds on the size and number of inputs of the circuits, and these are seen to satisfy the requirements of Corollary 6.4. The resulting threshold circuits treat their inputs as rational numbers represented by pairs of integers. These circuits can be evaluated at the given integer inputs by representing the integer a with the pair $(a, 1)$. The output of these circuits is the value of the resultant, represented as a rational number (a, b).

  Because the resultant is a polynomial with integer coefficients, its evaluation at an integer-valued point is itself an integer. This implies that a/b is an integer, so b necessarily divides a. To compute the binary representation of a/b, we compose the family of threshold circuits for the resultant with a circuit for integer division (item 3 of Theorem 2.12) to obtain a

circuit family that computes the integer representation of the resultant. The conclusion of Lemma 6.1 provides bounds on the bit complexity of a and b, and therefore the requirements to invoke Lemma 6.5 with the circuits for division are satisfied. By Lemma 6.5, because the language corresponding to a rational representation of the resultant is in CH, so is the language corresponding to the integer representation of the resultant.

- $R = \mathbb{Z}[y_1, \ldots, y_k]$: Corollary 5.13 constructs a polylogtime-uniform family of arithmetic circuits that take as input polynomials $F_0, \ldots, F_n \in \mathbb{Q}[y_1, \ldots, y_k][x_0, \ldots, x_n]$ and outputs the coefficient of the resultant $\mathrm{Res}(F_0, \ldots, F_n)$, which is itself a polynomial in $\mathbb{Q}[y_1, \ldots, y_k]$. If the polynomials F_i have degree at most d in the $\mathbf{x}$ variables and degree at most D in the $\mathbf{y}$ variables, then these circuits have size bounded by $(ndD)^{\mathrm{poly}(n,k)}$, which is at most singly-exponential in terms of the number of inputs. Applying the multi-output analogue of Corollary 6.4 to these arithmetic circuits allows us to decide the language corresponding to the resultant over $\mathbb{Q}[y_1, \ldots, y_k]$ in CH. As above, Corollary 5.13 provides bounds on size and number of inputs of the circuits, which are seen to satisfy the requirements in Corollary 6.4. The resulting threshold circuits can be evaluated at inputs from $\mathbb{Z}[y_1, \ldots, y_k]$ by representing the integer a with $(a, 1)$ as in the previous case.

  Similar to the integer case, when the input $F_0, \ldots, F_n$ are elements of $\mathbb{Z}[y_1, \ldots, y_k][x_0, \ldots, x_n]$, the resultant $\mathrm{Res}(F_0, \ldots, F_n)$ is a polynomial with integer coefficients, rather than rational coefficients. Therefore, we can use Lemma 6.6 to compose the family of threshold circuits for the resultant with circuits for integer division (item 3 of Theorem 2.12) applied to each coefficient to obtain integer representations of the coefficients of $\mathrm{Res}(F_0, \ldots, F_n)$. By Lemma 6.6, since the language corresponding to the resultant over $\mathbb{Q}[y_1, \ldots, y_k]$ is in CH, so is the language corresponding to these composed functions that compute the integer resultant.

- $R = \mathbb{F}_p$: Denote the input polynomials by $F_0, \ldots, F_n \in \mathbb{F}_p[\mathbf{x}]$. We can lift these to a collection of polynomials $\widehat{F}_0, \ldots, \widehat{F}_n \in \mathbb{Z}[\mathbf{x}]$ with integer coefficients by lifting each element of $\mathbb{F}_p$ to an integer between 0 and $p-1$ inclusive. Using the result from the case $R = \mathbb{Z}$, we can decide the value of any bit of $\mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n)$ in CH. The resultant $\mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n)$ is an integer. The fact that the resultant over $\mathbb{F}_p$ is the image of the resultant over $\mathbb{Z}$ under the map $\mathbb{Z} \to \mathbb{F}_p$ (see Definition 4.1) means that

$$\mathrm{Res}(F_0, \ldots, F_n) = \mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n) \bmod p.$$

  Thus, it remains to compute the remainder of $\mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n)$ modulo p. By item 3 of Theorem 2.12, there are logtime-uniform TC^0 circuits for integer division with remainder. In particular, there are logtime-uniform TC^0 circuits that compute the remainder of an integer when divided by p. We can compose these with the circuits that compute the resultant of $\widehat{F}_0, \ldots, \widehat{F}_n$ using Lemma 6.5. Since the language corresponding to the latter is in CH, so is the language corresponding to the composed functions, which is exactly the resultant over $\mathbb{F}_p$.

- $R = \mathbb{F}_p[y_1, \ldots, y_k]$: We reduce to the case of $R = \mathbb{Z}[y_1, \ldots, y_k]$ in a manner completely analogous to the case of $R = \mathbb{F}_p$. Let $F_0, \ldots, F_n \in \mathbb{F}_p[y_1, \ldots, y_k][\mathbf{x}]$ be the input polynomials, and let $\widehat{F}_0, \ldots, \widehat{F}_n \in \mathbb{Z}[\mathbf{y}][\mathbf{x}]$ be their lifts to the integers. Then we have

$$\mathrm{Res}(F_0, \ldots, F_n) = \mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n) \bmod p,$$

  so we can compute $\mathrm{Res}(F_0, \ldots, F_n)$ by first computing $\mathrm{Res}(\widehat{F}_0, \ldots, \widehat{F}_n)$ and then reducing each coefficient in this resultant modulo p as in the case of $R = \mathbb{F}_p$ above, using Lemma 6.6.

- $R = \mathbb{F}_{p^a}$: Recall that $\mathbb{F}_{p^a}$ is isomorphic to $\mathbb{F}_p[z]/(g(z))$, where $g \in \mathbb{F}_p[z]$ is a degree-a irreducible polynomial. Let $F_0, \ldots, F_n \in \mathbb{F}_{p^a}[\mathbf{x}]$ be the input polynomials, and let $\widehat{F}_0, \ldots, \widehat{F}_n \in \mathbb{Z}[z][\mathbf{x}]$ be their lifts to $\mathbb{Z}[z]$. Let $\hat{g}$ be a monic lift of $g(z)$ to $\mathbb{Z}[z]$. As in previous cases, by the universality of the resultant (see Definition 4.1), we have

  $$\operatorname{Res}(F_0, \ldots, F_n) = \operatorname{Res}(\widehat{F}_0, \ldots, \widehat{F}_n) \bmod (p, \hat{g}(z)).$$

  By the case $R = \mathbb{Z}[z]$, we can compute $\operatorname{Res}(\widehat{F}_0, \ldots, \widehat{F}_n)$ in CH, which is an element of $\mathbb{Z}[z]$. We use the circuit for pseudodivision over $\mathbb{Z}$ (item 4 of Theorem 2.12) combined with Lemma 6.5 to pseudodivide this polynomial by $\hat{g}$. Since $\hat{g}$ is monic, the pseudoquotient is the actual quotient. We then reduce the coefficients modulo p using circuits for integer division as in previous cases to obtain the resultant $\operatorname{Res}(F_0, \ldots, F_n)$.

- $R = \mathbb{F}_{p^a}[y_1, \ldots, y_k]$: We reduce to the case of $R = \mathbb{Z}[\mathbf{y}]$ as above by lifting the inputs. We then compute the resultant, pseudodivide each coefficient of $\mathbf{y}$ by a lift of $\hat{g}$, and then reduce every coefficient modulo p. □

By invoking the reduction of Proposition 4.4 from deciding the Nullstellensatz to computing the resultant, we obtain a CH procedure to decide satisfiability of systems of polynomial equations. To handle the case where the inputs have coefficients in a number field, we lift to the rational numbers using the following lemma.

Lemma 6.8. *Let $\mathbb{K} \coloneqq \mathbb{Q}[\alpha]$ be a number field and let $g \in \mathbb{Q}[z]$ be the minimal polynomial of α. Let $f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n]$ be a set of polynomials. Define $f'_1, \ldots, f'_m \in \mathbb{Q}[x_1, \ldots, x_n, z]$ to be the polynomials obtained from $f_1, \ldots, f_m$ by writing each $\mathbb{K}$-coefficient as a polynomial in $\mathbb{Q}[z]$ of degree less than $\deg g$. Then $\mathrm{V}(f_1, \ldots, f_m) \neq \varnothing$ if and only if $\mathrm{V}(f'_1, \cdots, f'_m, g) \neq \varnothing$. Further, if the former variety is zero-dimensional and has r points, then the latter variety is zero-dimensional and has $r \cdot \deg g$ points.*

Proof. The rings $\mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_m)$ and $\mathbb{Q}[\mathbf{x}, z]/(f'_1, \ldots, f'_m, g)$ are isomorphic. By the Nullstellensatz, $\mathrm{V}(f_1, \cdots, f_m) \neq \varnothing$ if and only if $\mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_m)$ is not the zero ring. Similarly, $\mathrm{V}(f'_1, \cdots, f'_m, g) \neq \varnothing$ if and only if $\mathbb{Q}[\mathbf{x}, z]/(f'_1, \ldots, f'_m, g)$ is not the zero ring. The first statement follows directly from these three facts.

Assume now that $\mathrm{V}(f_1, \ldots, f_m)$ is a zero-dimensional variety. The size of this variety, which we denote by r, is exactly the dimension of $\mathbb{K}[\mathbf{x}]/\operatorname{rad}(f_1, \ldots, f_m)$ as a $\mathbb{K}$-vector space. Let $\mathbb{L}$ be a splitting field of g. Let $\alpha_1, \ldots, \alpha_e$ be the roots of g in $\mathbb{L}$, where $e = \deg g$ and $\alpha = \alpha_1$. The number of points in $\mathrm{V}(f'_1, \ldots, f'_m, g)$ with last coordinate α_1 is exactly r by definition.

Let $\sigma_1, \ldots, \sigma_e$ be automorphisms of $\mathbb{L}$ such that $\sigma_i(\alpha) = \alpha_i$ for every i. Define $\mathbb{K}_i \coloneqq \sigma_i(\mathbb{K})$. The isomorphism σ_i induces an isomorphism $\mathbb{K}[\mathbf{x}]/\operatorname{rad}(f_1, \ldots, f_m) \cong \mathbb{K}_i[\mathbf{x}]/\operatorname{rad}(\sigma_i(f_1), \ldots, \sigma_i(f_m))$. Combined with the statements in the previous paragraph, this isomorphism shows that the number of points with last coordinate α_i in $\mathrm{V}(f'_1, \ldots, f'_m, g)$ is r for every i. This completes the proof of the second statement. □

We now combine Proposition 4.4 with Theorem 6.7 to prove that Hilbert's Nullstellensatz can be decided in CH.

Theorem 6.9. *Let R be one of the rings $\mathbb{Z}$, $\mathbb{Z}[y_1, \ldots, y_k]$, a number field $\mathbb{K}$, $\mathbb{K}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Then Hilbert's Nullstellensatz over R can be decided in CH.*

Proof. Let $f_1, \ldots, f_m \in R[x_1, \ldots, x_n]$ be the polynomials whose satisfiability we must decide. Let d and h be bounds on the degrees and heights, respectively, of the f_i. If R is either $\mathbb{K}$ or $\mathbb{K}[y_1, \ldots, y_k]$ then we use Lemma 6.8 to reduce to the case when $R = \mathbb{Q}$ or $\mathbb{Q}[y_1, \ldots, y_k]$, respectively, after which we pass to a common denominator, further reducing to the case $R = \mathbb{Z}$ or $R = \mathbb{Z}[y_1, \ldots, y_k]$. If R is one of the rings $\mathbb{F}_p, \mathbb{F}_{p^a}, \mathbb{F}_p[y_1, \ldots, y_k], \mathbb{F}_{p^a}[y_1, \ldots, y_k]$ and if p (or p^a) is smaller than $15nd^n$, then we use Lemma 2.13 to pass to an extension $\mathbb{F}_{p^b}$ or $\mathbb{F}_{p^b}[y_1, \ldots, y_k]$ such that $p^b \geq 15nd^n$. The degree of the extension required is polynomial in the input size, therefore doing so only increases the heights of the inputs by a polynomial factor. Similarly, arithmetic in the extension can be simulated efficiently using arithmetic in the original ring R. Further, satisfiability of the system is unchanged by passing to extensions.

Apply the randomized algorithm of Proposition 4.4 to the input $f_1, \ldots, f_m$ and denote the resulting polynomials by $G_{i,j} \in R[t, w, u, x_0, \ldots, x_n]$ where $i \in \{0, 1, \ldots, n\}$ and $j \in [n]$. To decide if the system $f_1 = \cdots = f_m = 0$ is satisfiable, we must decide if there is a $j \in [n]$ such that $(\mathrm{TT}_w \mathrm{TT}_t \operatorname{Res}(G_{0,j}, \ldots, G_{n,j}))(0) = 0$.

This second task can be solved in CH as follows. We iterate over all choices of $j \in [n]$. For each $j \in [n]$, we compute $R_j(t, w, u) \coloneqq \operatorname{Res}(G_{0,j}, \ldots, G_{n,j})$ in CH using the algorithm of Theorem 6.7. The resultant R_j is computed with respect to the $\mathbf{x}$ variables and is a polynomial in $R[t, w, u]$. Our task is to decide if $\mathrm{TT}_w \mathrm{TT}_t R_j$ has a zero constant term. We do this in CH as follows: we first nondeterministically guess the exponents of the trailing monomials of R_j with respect to t and w, and then verify that (1) the corresponding polynomial in u has a zero constant term, and (2) all smaller monomials in t and w have a coefficient of zero. The first verification task can be done in CH, since we can compute the coefficients of R_j in CH. The second can be done in coNP with a CH oracle, since we must verify that all smaller monomials have a zero coefficient, and the coefficient of any single monomial can be computed in CH. Since $\mathsf{coNP}^{\mathsf{CH}} = \mathsf{CH}$, this second verification task can likewise be performed in CH.

Thus, we have a randomized reduction from the task of deciding satisfiability of $f_1 = \cdots = f_m = 0$ to a problem that can be solved in CH. This reduction succeeds with probability at least $2/3$, so Hilbert's Nullstellensatz is in $\mathsf{BPP}^{\mathsf{CH}}$. Since $\mathsf{BPP} \subseteq \mathsf{PP} \subseteq \mathsf{CH}$, we can decide Hilbert's Nullstellensatz in CH as claimed. □

6.3 Counting solutions in zero-dimensional systems

In Section 4.3, we saw that to count solutions to zero-dimensional systems of equations, it is sufficient to compute multivariate resultants, univariate GCD's, and count the number of distinct roots of a univariate polynomial in the algebraic closure of its coefficient field. The results of Section 6.2 show that we can compute multivariate resultants in CH. In this section, we develop the additional machinery necessary to compute univariate GCD's and the number of distinct roots of a univariate polynomial. For our application to counting roots of zero-dimensional systems of equations, we need to perform these operations on polynomials of exponentially-large degree. The coefficients of these polynomials will be computable in CH, and our goal is to compute the coefficients of the GCD and the number of distinct roots in CH. To do this, it suffices, by Lemmas 6.5 and 6.6, to show that the GCD and number of distinct roots can be computed in polylogtime-uniform TC^0.

We start by observing that the resultant of two polynomials can be computed by polylogtime-uniform TC^0 circuits. This follows from Lemma 6.1 applied to the circuits we construct for the multivariate resultant of two polynomials.

Lemma 6.10. *Let R be one of the rings $\mathbb{Z}, \mathbb{Z}[y_1, \ldots, y_k], \mathbb{Q}, \mathbb{Q}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Then the resultant of two univariate polynomials with coefficients in R can be computed in polylogtime-uniform TC^0.*

Proof. The circuits we construct in Lemma 5.12 and Corollary 5.13 for the multivariate resultant are polynomial sized when $n = 2$. Applying Lemma 6.1 to these circuits immediately gives us polylogtime-uniform TC^0 circuits over $\mathbb{Q}$ and $\mathbb{Q}[\mathbf{y}]$ respectively. For the remaining rings, we use the same arguments as in the proof of Theorem 6.7, first lifting the input to $\mathbb{Z}$ or $\mathbb{Z}[y_1, \ldots, y_k]$, then computing the resultant, and finally projecting back down to the desired ring R. □

We note that constant-depth algebraic circuits for the resultant of two polynomials have been constructed prior to our work. Andrews and Wigderson [AW24] construct circuits for such resultants over $\mathbb{Q}$ using constant depth versions of the Girard–Newton identities, and Bhattacharjee, Kumar, Rai, Ramanathan, Saptharishi, and Saraf [BKR+25b] construct circuits over any field using Lagrange inversion. Neither of these works addressed the uniformity of their construction, but it is not hard to deduce uniformity of their constructions (in the case of the latter work, only when the field is $\mathbb{Q}$) using the results in Section 3.

Remark 6.11. It is not hard to show that the resultant of two univariate polynomials is complete for polylogtime-uniform TC^0. As Lemma 6.10 shows, the resultant of two polynomials can be computed in polylogtime-uniform TC^0. To show that the resultant is TC^0-hard, we reduce from the known TC^0-complete problem of integer powering: given an n-bit integer a and an $O(\log n)$-bit integer b, compute a^b. Hesse, Allender, and Mix Barrington [HAM02] showed that powering is complete for logtime-uniform TC^0. To reduce integer powering to the resultant, observe that if we regard the constant 1 as a polynomial of degree b, then

$$\operatorname{Res}(a, 1) = a^b.$$

In particular, to compute a^b, it suffices to compute the resultant of two univariate polynomials of degree at most $n^{O(1)}$. This proves that the resultant is hard for logtime-uniform TC^0. ◊

We now show how resultant computations can be used to compute greatest common divisors of univariate polynomials. The following reduction is from [BKR+25b].

Lemma 6.12 ([BKR+25b, Section 1.2])**.** *Let $\mathbb{F}$ be a field. If f and g are univariate polynomials in $\mathbb{F}[x]$, then*

$$\gcd(f, g)(y) = \frac{\operatorname{TT}_z \operatorname{Res}_x(z \cdot (y - x) + f(x), f(x) + u \cdot g(x))}{\operatorname{TT}_z \operatorname{Res}_x(z + f(x), f(x) + u \cdot g(x))}.$$

Proof. For any univariate polynomials $a, b, c \in \mathbb{F}[x]$, the Poisson formula implies that, up to a sign, we have

$$\operatorname{TT}_z \operatorname{Res}_x(zb + c, a) = a_0^{\max(\deg b, \deg c)} \left(\prod_{\alpha \in \mathrm{V}(a) \setminus \mathrm{V}(c)} c(\alpha)^{m(\alpha)} \right) \times \left(\prod_{\alpha \in \mathrm{V}(a) \cap \mathrm{V}(c)} b(\alpha)^{m(\alpha)} \right),$$

where z is a new variable, a_0 is the leading coefficient of a, and $m(\alpha)$ is the multiplicity of α as a root of a. As a consequence, we have, up to a sign,

$$\prod_{\alpha \in \mathrm{V}(a) \cap \mathrm{V}(c)} b(\alpha)^{m(\alpha)} = a_0^{\max(0, \deg b - \deg c)} \frac{\operatorname{TT}_z \operatorname{Res}_x(zb + c, a)}{\operatorname{TT}_z \operatorname{Res}_x(z + c, a)}.$$

Now consider the polynomials $f, g \in \mathbb{F}[x]$ and let u be a new variable. As a polynomial with coefficients in $\overline{\mathbb{F}(u)}$, the polynomial $f(x) + ug(x)$ factors into a product of linear forms. All common factors of $f(x)$ and $f(x) + ug(x)$ are necessarily factors of $\gcd(f, g)$. Moreover, these common factors have the same multiplicity in $f(x) + ug(x)$ as they do in $\gcd(f, g)$. The result now follows from the discussion above invoked in the field $\mathbb{F}(u, y)$, with $a(x) = f(x) + u \cdot g(x)$, $b(x) = (y - x)$, and $c(x) = f(x)$. □

By combining Lemma 6.10 with Lemma 6.12, we obtain a polylogtime-uniform family of TC^0 circuits to compute the GCD of two polynomials.

Lemma 6.13. *Let R be one of the rings $\mathbb{Z}$, $\mathbb{Z}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Then the greatest common divisor in the fraction field of R of two univariate polynomials with coefficients in R can be computed in polylogtime-uniform TC^0. The coefficients of the GCD will be represented as b_i/c with $b_i, c \in R$.*

Proof. Suppose f and g are the inputs. We compute resultants $r_1 \coloneqq \operatorname{Res}_x(z \cdot (y - x) + f, f + u \cdot g)$ and $r_2 \coloneqq \operatorname{Res}_x(z + f, f + u \cdot g)$ by invoking Lemma 6.10 for the ring $R[z, u, y]$. From r_1, r_2, the trailing terms $a_1 \coloneqq \operatorname{TT}_z r_1$ and $a_2 \coloneqq \operatorname{TT}_z r_2$ can be computed by inspection of the coefficients as follows. For each $0 \leq i \leq \deg(a_1)$, we first determine if the coefficient of z^i in r_1 corresponds to $\operatorname{TT}_z r_1$ by checking that some monomial $z^i u^j y^k$ has a nonzero coefficient in r_1 and that all monomials of the form $z^{i'} u^j y^k$ for $i' < i$ have coefficients of zero. We then compute the bits of the coefficient of the monomial $u^j y^k$ in $\operatorname{TT}_z r_1$: the ℓ^{th} bit of this coefficient is 1 exactly when there is some i such that (1) the coefficient of z^i corresponds to $\operatorname{TT}_z r_1$, and (2) the ℓ^{th} bit of the coefficient of $z^i u^j y^k$ in r_1 is a 1. This computation can be carried out by polynomial-size constant-depth boolean circuits built from AND and OR gates in a straightforward manner. An entirely analogous computation produces the coefficients of $a_2 = \operatorname{TT}_z r_2$.

We have $a_1 \in R[u, y]$ and $a_2 \in R[u]$. By Lemma 6.12, we know that a_2 divides a_1, that the quotient is independent of u, and that the quotient is the GCD of f and g. Suppose $a_1 = \sum_i a_{1,i} y^i$ with $a_{1,i} \in R[u]$. The only way for a_1/a_2 to be independent of u is for each $a_{1,i}$ to be a multiple of a_2 in the fraction field of R. This multiple can therefore be read off of just the leading terms; that is, $a_{1,i}/a_2$ is simply the leading coefficient of $a_{1,i}$ divided by that of a_2. Setting c to be the leading coefficient of a_2 and b_i to be the leading coefficient of $a_{1,i}$ therefore gives us the GCD in the required form. □

Next, we show how resultant computations can be used to compute factors of a univariate polynomial f that correspond to roots that occur with multiplicity at least k. This computation is inspired by the squarefree decomposition algorithm in [AW24]. For a univariate polynomial f with coefficients in a field $\mathbb{F}$, we define $f_{>k}$ as

$$f_{>k}(x) \coloneqq \prod_{\alpha \in \mathrm{V}(f), m(\alpha) > k} (y - \alpha)^{m(\alpha)},$$

the product of all factors of multiplicity greater than k.

Lemma 6.14. *Let $\mathbb{F}$ be a field. If f is a polynomial in $\mathbb{F}[x]$, then we have*

$$f_{>k} = c' \cdot \frac{\operatorname{TT}_z \operatorname{Res}_x\big(z \cdot (y - x) + \mathrm{D}(f) + v\,\mathrm{D}^2(f) + \cdots + v^{k-1}\,\mathrm{D}^k(f), f\big)}{\operatorname{TT}_z \operatorname{Res}_x\big(z + \mathrm{D}(f) + v\,\mathrm{D}^2(f) + \cdots + v^{k-1}\,\mathrm{D}^k(f), f\big)},$$

where $\mathrm{D}^i(f)$ is the i^{th} Hasse derivative of f and $c' \in \mathbb{F} \setminus \{0\}$.

Proof. We invoke the equation

$$\prod_{\alpha \in \mathrm{V}(a) \cap \mathrm{V}(c)} b(\alpha)^{m(\alpha)} = a_0^{\max(0, \deg b - \deg c)} \frac{\operatorname{TT}_z \operatorname{Res}_x(zb + c, a)}{\operatorname{TT}_z \operatorname{Res}_x(z + c, a)},$$

with $a(x) = f(x)$, $b(x) = (y - x)$, and $c(x) = \mathrm{D}(f) + v\,\mathrm{D}^2(f) + \cdots + v^{k-1}\,\mathrm{D}^k(f)$, where v is a new variable. Any $\alpha \in \mathrm{V}(f) \cap \mathrm{V}(c)$ is necessarily a root of $\mathrm{D}(f), \ldots, \mathrm{D}^k(f)$. The roots of f that are also roots of these Hasse derivatives are exactly those roots of f that occur with multiplicity more than k. Thus, $f_{>k}(y)$ is precisely $\prod_{\alpha \in \mathrm{V}(f) \cap \mathrm{V}(c)} (y - \alpha)^{m(\alpha)}$ as claimed. □

We also define $f_{=k}$ as $f_{>k-1}(x)/f_{>k}$. If $f_{=k}$ has degree r, then $f(x)$ has exactly r/k distinct roots that occur with multiplicity exactly k. Observe that the denominator in the expression in Lemma 6.14 is independent of y, so to compute the y-degree of $f_{>k}$, it suffices to compute the numerator. Using this observation, we design a polylogtime-uniform family of TC^0 circuits to compute the number of distinct roots of a univariate polynomial.

Lemma 6.15. *Let R be one of the rings $\mathbb{Z}$, $\mathbb{Z}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Let $\mathbb{K}$ be the algebraic closure of the field of fractions of R. Then the number of distinct roots in $\mathbb{K}$ of a univariate polynomial with coefficients in R can be computed in polylogtime-uniform TC^0.*

Proof. Let $f \in R[x]$ be the input polynomial. First, we compute all the Hasse derivatives $\mathrm{D}^i(f)$ for $1 \leq i \leq \deg f$ in parallel. This involves only integer arithmetic. Then we compute the resultants $r_k \coloneqq \mathrm{TT}_z \operatorname{Res}_x\big(z \cdot (y - x) + \mathrm{D}(f) + v\,\mathrm{D}^2(f) + \cdots + v^{k-1}\,\mathrm{D}^k(f), f\big)$ using the circuit from Lemma 6.10 for the ring $R[z, y, v]$. By Lemma 6.14 and the discussion following it, the y-degree of r_k is exactly the y-degree of $f_{>k}$. Once these degrees are computed for all k up to $\deg f$, we can compute

$$\deg f - \deg f_{>1} + \frac{\deg f_{>1} - \deg f_{>2}}{2} + \cdots + \frac{\deg f_{>d-1}}{d}$$

using circuits for integer arithmetic, and this is the desired number of distinct roots of f in $\mathbb{K}$. □

Now that we have polylogtime-uniform TC^0 algorithms to compute GCD's and the number of roots, we are ready to state the main result of this subsection. Using Proposition 4.12, we obtain a CH algorithm to count the number of solutions to a zero-dimensional system of polynomial equations.

Theorem 6.16. *Let R be one of the rings $\mathbb{Z}$, $\mathbb{Z}[y_1, \ldots, y_k]$, a number field $\mathbb{K}$, $\mathbb{K}[y_1, \ldots, y_k]$, $\mathbb{F}_p$, $\mathbb{F}_p[y_1, \ldots, y_k]$, $\mathbb{F}_{p^a}$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, where p is a prime number. Let $\mathbb{L}$ be the algebraic closure of the field of fractions of R. There is a $\mathsf{FP}^{\mathsf{CH}}$ algorithm that counts the number of solutions in $\mathbb{L}^n$ to zero-dimensional systems of polynomial equations with coefficients in R.*

Proof. Let $f_1, \ldots, f_m \in R[x_1, \ldots, x_n]$ be the polynomials whose roots we must count. Let d and h be bounds on the degrees and heights, respectively, of the f_i. If R is either $\mathbb{K}$ or $\mathbb{K}[y_1, \ldots, y_k]$, then we use Lemma 6.8 to reduce to the case when $R = \mathbb{Q}$ or $\mathbb{Q}[y_1, \ldots, y_k]$, respectively, and then subsequently pass to a common denominator to reduce to the case where $R = \mathbb{Z}$ or $R = \mathbb{Z}[y_1, \ldots, y_k]$, since scaling by field elements does not change the number of roots. If R is one of the rings $\mathbb{F}_p$, $\mathbb{F}_{p^a}$, $\mathbb{F}_p[y_1, \ldots, y_k]$, or $\mathbb{F}_{p^a}[y_1, \ldots, y_k]$, and if p (or p^a) is smaller than $100nd^{2n}$, then we use Lemma 2.13 to pass to an extension $\mathbb{F}_{p^b}$ or $\mathbb{F}_{p^b}[y_1, \ldots, y_k]$ such that $p^b \geq 100nd^{2n}$. The degree of the extension required is polynomial in the input size, therefore doing so only increases the heights of the inputs by a polynomial factor. Similarly, arithmetic in the extension can be simulated efficiently using arithmetic in the original ring R. Further, the number of roots is unchanged by passing to extensions.

Apply the randomized algorithm of Proposition 4.12 to the input $f_1, \ldots, f_m$ and denote the resulting polynomials by $G_{i,j} \in R[t, u, x_0, \ldots, x_n]$ where $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, 2\}$. To count the number of roots of $f_1 = \cdots = f_m = 0$, we must count the number of distinct roots of the GCD of $\mathrm{TP}_u\,\mathrm{TT}_t \operatorname{Res}(G_{0,1}, \ldots, G_{n,1})$ and $\mathrm{TP}_u\,\mathrm{TT}_t \operatorname{Res}(G_{0,2}, \ldots, G_{n,2})$.

This task can be solved in CH as follows. For $j = 1, 2$, we compute $R_j(t, u) \coloneqq \operatorname{Res}(G_{0,j}, \ldots, G_{n,j})$ in CH using the algorithm of Theorem 6.7. The resultant R_j is computed with respect to the $\mathbf{x}$ variables and is a polynomial in $R[t, u]$. We first nondeterministically guess the exponents of the trailing monomials of R_j with respect to t, and then verify all smaller monomials in t have a coefficient of zero. This can be done in coNP with a CH oracle, since coefficient in u of the coefficients

of any single monomial in t can be computed in CH. Since $\mathsf{coNP}^{\mathsf{CH}} = \mathsf{CH}$, this verification task can be performed in CH.

We then nondeterministically guess the exponent of the trailing monomial of u in $\mathrm{TT}_t R_j$. This again can be done in CH. We then have access to the polynomials $\mathrm{TP}_u \mathrm{TT}_t R_j$ in the form of CH oracles for the coefficients of each monomial. Using Lemma 6.5 and Lemma 6.13, we can obtain CH oracles to the coefficients of the GCD of these two polynomials. These coefficients will be represented as ratios with a common denominator, but we can ignore the denominator since scaling will not affect the next step. Finally, using Lemma 6.5 once more with Lemma 6.15, we can count the number of distinct roots, which is the same as the number of roots of the original system.

Thus, we have a randomized reduction from the task of counting the number of roots of $f_1 = \cdots = f_m = 0$ to a problem that can be solved in $\mathsf{FP}^{\mathsf{CH}}$. This reduction succeeds with probability at least $2/3$, so the counting version of Hilbert's Nullstellensatz is in the functional version of $\mathsf{BPP}^{\mathsf{CH}}$, which in turn lies in $\mathsf{FP}^{\mathsf{CH}}$. □

7 Applications of the Nullstellensatz

In this section, we give a few examples of problems that can be reduced to Hilbert's Nullstellensatz. These problems were previously only known to be in PSPACE, and our results show that they can in fact be solved in CH. This list is far from exhaustive. In this section, we let $\mathbb{F}$ denote any field for which our results hold, namely $\mathbb{Q}$, a number field $\mathbb{K}$, a finite field $\mathbb{F}_q$, or a function field over one of these fields. In the function field case we assume that the inputs are restricted to polynomials instead of arbitrary rational functions.

The first application is to deciding radical ideal membership in the algebraic closure. Equivalently, this is the problem of testing if a given polynomial vanishes on a given variety.

Corollary 7.1. *Given polynomials $g, f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$, deciding if*

$$g \in \operatorname{rad}\big((f_1, \ldots, f_m)\overline{\mathbb{F}}[x_1, \ldots, x_n]\big)$$

can be done in CH.

Proof. Consider the system of equations $f_1 = \cdots = f_m = 1 - zg = 0$, where z is a new variable. By the Nullstellensatz, the condition that $g \in \operatorname{rad}\big((f_1, \ldots, f_m)\overline{\mathbb{F}}[x_1, \ldots, x_n]\big)$ is equivalent to the condition that this system is not satisfiable. This is usually called the Rabinowitsch trick. Satisfiability can be decided in CH, and therefore so can membership in the ideal $\operatorname{rad}\big((f_1, \ldots, f_m)\overline{\mathbb{F}}[x_1, \ldots, x_n]\big)$. □

The second problem is that of computing the dimension of an algebraic variety given its defining equations. To make this a decision problem, our algorithms will also take as input a number and will decide if the dimension equals this number. A similar reduction between computing the dimension of a variety and Hilbert's Nullstellensatz was given by Koiran [Koi97].

Corollary 7.2. *Given polynomials $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ and an integer r, deciding if*

$$\dim \mathrm{V}(f_1, \ldots, f_m) = r$$

can be done in CH.

Proof. We design a randomized Turing reduction to satisfiability. Let $d \coloneqq \max \deg f_i$ and let $B \subseteq \mathbb{F}$ be a subset of size at least $100n^2d^n$. If $\mathbb{F}$ is not big enough to pick this subset, we pass to a sufficiently-large field extension using Lemma 2.13. Let $\ell_1, \ldots, \ell_n$ be random linear polynomials whose coefficients

are picked uniformly and independently from B. For any integer k, if $\dim \mathrm{V}(f_1, \ldots, f_m) \geq k$, then with probability at least $1 - 2nd^n/|B|$, the system $f_1 = \cdots = f_m = \ell_1 = \cdots = \ell_k = 0$ is satisfiable (Lemma 4.9). Conversely, if $\dim \mathrm{V}(f_1, \ldots, f_m) < k$, then with the same probability, the system $f_1 = \cdots = f_m = \ell_1 = \cdots = \ell_k = 0$ is unsatisfiable. Therefore, based on the satisfiability of these systems for all $k \in [n]$, we can compute the dimension of $\mathrm{V}(f_1, \ldots, f_m)$ and decide if this equals r. □

The final application we discuss is the computation of the rank of a tensor over $\overline{\mathbb{F}}$. We are given a tensor $T \in \mathbb{F}^{d_1} \otimes \cdots \otimes \mathbb{F}^{d_k}$ and a natural number $r \in \mathbb{N}$ as input, and we must decide if T the rank of T is equal to r. It is straightforward to write down a system of polynomial equations that is solvable if and only if T has rank at most r. Since Theorem 6.9 allows us to decide the solvability of this system of equations over the algebraic closure $\overline{\mathbb{F}}$ in CH, we can likewise decide in CH whether T has rank r when viewed as a tensor over the algebraic closure $\overline{\mathbb{F}}$.

Corollary 7.3. *Given an integer r and a tensor $T \in \mathbb{F}^{d_1} \otimes \cdots \otimes \mathbb{F}^{d_k}$ explicitly as a list of entries, deciding if the rank of T as an element of $\overline{\mathbb{F}}^{d_1} \otimes \cdots \otimes \overline{\mathbb{F}}^{d_k}$ is exactly r is in CH.*

Proof. Every tensor in $\mathbb{F}^{d_1} \otimes \cdots \otimes \mathbb{F}^{d_k}$ has rank at most $d_1 \cdots d_k$, so if $r > d_1 \cdots d_k$, we immediately reject the input. Otherwise, let $x_{i,j,p}$ be a set of variables where $i \in [k]$, $j \in [d_i]$, and $p \in [r]$. For each index $(j_1, \ldots, j_k) \in [d_1] \times \cdots \times [d_k]$, we write the polynomial equation

$$T_{j_1,\ldots,j_k} = \sum_{p=1}^{r} \prod_{i=1}^{k} x_{i,j_i,p}.$$

Taken together, these equations express that the tensor T can be written as the sum of r rank-one tensors, where the p^{th} such tensor is the outer product of the vectors $\mathbf{x}_{1,\bullet,p} \otimes \cdots \otimes \mathbf{x}_{k,\bullet,p}$ for $\mathbf{x}_{i,\bullet,p} = (x_{i,1,p}, \ldots, x_{i,d_i,p})$. It follows that this system of equations is satisfiable over $\overline{\mathbb{F}}$ if and only if T has rank at most r as a tensor over $\overline{\mathbb{F}}$.

This is a collection of at most $d_1 \cdots d_k$ equations in at most $k \cdot (d_1 \cdots d_k)^2$ variables. Since the tensor T is given as a list of $d_1 \cdots d_k$ field elements, we can write down this system of equations in time that is polynomial in the size of the input. Applying the algorithm of Theorem 6.9 allow us to decide if T has rank at most r in CH. To decide if the rank of T is exactly r, we use the same procedure to test if T has rank at most $r - 1$, again in CH. □

References

[AAD00] Manindra Agrawal, Eric Allender, and Samir Datta. "On TC0, AC0, and Arithmetic Circuits". In: *Journal of Computer and System Sciences* 60.2 (2000), pp. 395–421. ISSN: 0022-0000. DOI: 10.1006/jcss.1999.1675 (cit. on p. 42).

[ABKPM09] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. "On the Complexity of Numerical Analysis". In: *SIAM Journal on Computing* 38.5 (2009), pp. 1987–2006. DOI: 10.1137/070697926 (cit. on p. 45).

[ABNSW25] Rida Ait El Manssour, Nikhil Balaji, Klara Nosan, Mahsa Shirmohammadi, and James Worrell. "A parametric version of the Hilbert Nullstellensatz". In: *2025 Symposium on Simplicity in Algorithms (SOSA)*. 2025, pp. 444–451. DOI: 10.1137/1.9781611978315.32 (cit. on p. 2).

[AiK81] L. A. A˘ izenberg and A. M. Kytmanov. "Multidimensional analogues of Newton's formulas for systems of nonlinear algebraic equations and some of their applications". In: *Sibirsk. Mat. Zh.* 22.2 (1981), pp. 19–30, 235. ISSN: 0037-4474 (cit. on p. 7).

[AKRRV01] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. "Time-space tradeoffs in the counting hierarchy". In: *Proceedings of the 16th Annual IEEE Conference on Computational Complexity (CCC 2001)*. 2001, pp. 295–302. DOI: 10.1109/CCC.2001.933896 (cit. on p. 42).

[AW24] Robert Andrews and Avi Wigderson. "Constant-Depth Arithmetic Circuits for Linear Algebra Problems". In: *Proceedings of the 65th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2024)*. 2024. DOI: 10.1109/FOCS61266.2024.00138. arXiv: 2404.10839 [cs.CC] (cit. on pp. 5, 7, 9, 23, 52, 53).

[AW90] Eric Allender and Klaus W. Wagner. "Counting Hierarchies: Polynomial Time and Constant Depth Circuits". In: *Bull. EATCS* 40 (1990), pp. 182–194 (cit. on p. 3).

[AY83] L. A. Aizenberg and A. P. Yuzhakov. *Integral Representations and Residues in Multidimensional Complex Analysis*. Vol. 58. Translations of Mathematical Monographs. Providence, R.I.: American Mathematical Society, 1983. ISBN: 0821815504 (cit. on pp. 32, 36).

[BC06] Peter Bürgisser and Felipe Cucker. "Counting complexity classes for numeric computations II: Algebraic and semialgebraic sets". In: *Journal of Complexity* 22.2 (2006), pp. 147–191. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2005.11.001 (cit. on pp. 2, 4).

[BCL05] Peter Bürgisser, Felipe Cucker, and Martin Lotz. "Counting Complexity Classes for Numeric Computations. III: Complex Projective Sets". In: *Found. Comput. Math.* 5.4 (2005), pp. 351–387. DOI: 10.1007/S10208-005-0146-X (cit. on p. 4).

[BCS97] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. With the collaboration of Thomas Lickteig. Springer-Verlag, Berlin, 1997, pp. xxiv+618. ISBN: 3-540-60582-7. DOI: 10.1007/978-3-662-03338-8 (cit. on pp. 25, 29).

[BCSS98] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. With a foreword by Richard M. Karp. Springer-Verlag, New York, 1998, pp. xvi+453. ISBN: 0-387-98281-7. DOI: 10.1007/978-1-4612-0701-6 (cit. on p. 2).

[BKR+25a] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Closure under factorization from a result of Furstenberg*. Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR25-084. 2025 (cit. on p. 7).

[BKR+25b] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Constant-depth circuits for polynomial GCD over any characteristic*. Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR25-085. 2025 (cit. on pp. 23, 52).

[BL07] Peter Bürgisser and Martin Lotz. "The Complexity of Computing the Hilbert Polynomial of Smooth Equidimensional Complex Projective Varieties". In: *Found. Comput. Math.* 7.1 (2007), pp. 59–86. DOI: 10.1007/S10208-005-0175-0 (cit. on p. 4).

[Bor77] Allan Borodin. "On Relating Time and Space to Size and Depth". In: *SIAM Journal on Computing* 6.4 (1977), pp. 733–744. DOI: 10.1137/0206054 (cit. on p. 2).

[Bro87] W. Dale Brownawell. "Bounds for the Degrees in the Nullstellensatz". In: *Annals of Mathematics* 126.3 (1987), pp. 577–591. ISSN: 0003486X, 19398980. DOI: 10.2307/1971361 (cit. on p. 1).

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines". In: *Bull. Amer. Math. Soc. (N.S.)* 21.1 (1989), pp. 1–46. ISSN: 0273-0979,1088-9485. DOI: 10.1090/S0273-0979-1989-15750-9 (cit. on p. 2).

[BvH82] Allan Borodin, Joachim von zur Gathen, and John Hopcroft. "Fast parallel matrix and GCD computations". In: *Information and Control* 52.3 (1982), pp. 241–256. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(82)90766-5 (cit. on p. 2).

[Can90] John Canny. "Generalised characteristic polynomials". In: *Journal of Symbolic Computation* 9.3 (1990). Computational algebraic complexity editorial, pp. 241–250. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(08)80012-0 (cit. on pp. 8, 24).

[CGH88] Léandro Caniglia, André Galligo, and Joos Heintz. "Borne simple exponentielle pour les degrés dans le théorème des zéros sur un corps de caractéristique quelconque". In: *C. R. Acad. Sci. Paris Sér. I Math.* 307.6 (1988), pp. 255–258 (cit. on p. 1).

[Cha02] Charalambos A. Charalambides. *Enumerative combinatorics*. CRC Press Series on Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, FL, 2002, pp. xvi+609. ISBN: 1-58488-290-5 (cit. on p. 16).

[CLO05] David A. Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. 2nd ed. Springer New York, NY, 2005. DOI: 10.1007/b138611 (cit. on pp. 22, 23).

[Csa76] L. Csanky. "Fast Parallel Matrix Inversion Algorithms". In: *SIAM Journal on Computing* 5.4 (1976), pp. 618–623. DOI: 10.1137/0205040 (cit. on pp. 2, 5).

[DPR61] Martin Davis, Hilary Putnam, and Julia Robinson. "The decision problem for exponential diophantine equations". In: *Ann. of Math. (2)* 74 (1961), pp. 425–436. ISSN: 0003-486X. DOI: 10.2307/1970289 (cit. on p. 1).

[EFP98] Alfredo Eisinberg, Giuseppe Franzé, and Paolo Pugliese. "Vandermonde matrices on integer nodes". In: *Numerische Mathematik* 80.1 (1998), pp. 75–85 (cit. on p. 16).

[Eis13] David Eisenbud. *Commutative algebra: with a view toward algebraic geometry*. Vol. 150. Springer Science & Business Media, 2013 (cit. on p. 36).

[FG90] Noaï Fitchas and André Galligo. "Nullstellensatz effectif et conjecture de Serre (théorème de Quillen-Suslin) pour le calcul formel". In: *Math. Nachr.* 149 (1990), pp. 231–253. DOI: 10.1002/mana.19901490118 (cit. on p. 1).

[For24] Michael A. Forbes. "Low-Depth Algebraic Circuit Lower Bounds over Any Field". In: *39th Computational Complexity Conference (CCC 2024)*. Ed. by Rahul Santhanam. Vol. 300. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 31:1–31:16. ISBN: 978-3-95977-331-7. DOI: 10.4230/LIPIcs.CCC.2024.31 (cit. on p. 23).

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. "Parity, Circuits, and the Polynomial-Time Hierarchy". In: *Math. Syst. Theory* 17.1 (1984), pp. 13–27. DOI: 10.1007/BF01744431 (cit. on p. 6).

[Gat86] Joachim von zur Gathen. "Parallel arithmetic computations: A survey". In: *Proceedings of the 11th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1986)*. Ed. by Jozef Gruska, Branislav Rovan, and Juraj Wiedermann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 93–112. ISBN: 978-3-540-39909-4. DOI: 10.1007/BFb0016236 (cit. on p. 12).

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge University Press, 2013 (cit. on pp. 15, 16).

[GKZ94] Israel M. Gelfand, Mikhail M. Kapranov, and Andrei V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. 1st ed. Birhaäuser Boston, MA, 1994, pp. x+523. ISBN: 978-0-8176-4771-1. DOI: 10.1007/978-0-8176-4771-1 (cit. on p. 22).

[HAM02] William Hesse, Eric Allender, and David A. Mix Barrington. "Uniform constant-depth threshold circuits for division and iterated multiplication". In: *Journal of Computer and System Sciences* 65.4 (2002). Special Issue on Complexity 2001, pp. 695–716. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(02)00025-9 (cit. on pp. 5, 6, 14, 52).

[Hei83] J. Heintz. "Definability and fast quantifier elimination in algebraically closed fields". In: *Theor. Comput. Sci.* 24 (1983), pp. 239–277 (cit. on p. 27).

[Her26] Grete Hermann. "Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. (Unter Benutzung nachgelassener Sätze von K. Hentzelt)". In: *Mathematische Annalen* 95 (1926), pp. 736–788. DOI: 10.1007/BF01206635 (cit. on p. 1).

[Hil02] David Hilbert. "Mathematical problems". In: *Bull. Amer. Math. Soc.* 8.10 (1902), pp. 437–479. ISSN: 0002-9904. DOI: 10.1090/S0002-9904-1902-00923-3 (cit. on p. 1).

[HJSS02] J. Heintz, G. Jeronimo, J. Sabia, and P. Solerno. *Intersection theory and deformation algorithms: the multi-homogeneous case*. Manuscript. 2002 (cit. on p. 32).

[Ier89a] Doug Ierardi. "Quantifier Elimination in the Theory of an Algebraically-closed Field". In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1989)*. Ed. by David S. Johnson. ACM, 1989, pp. 138–147. DOI: 10.1145/73007.73020 (cit. on pp. 8, 24, 25).

[Ier89b] Doug J Ierardi. *The complexity of quantifier elimination in the theory of an algebraically closed field*. Tech. rep. Cornell University, 1989 (cit. on p. 24).

[IK93] Douglas Ierardi and Dexter Kozen. "Parallel Resultant Computation". In: *Synthesis of Parallel Algorithms*. Ed. by John H. Reif. Morgan Kaufmann Publishers Inc., 1993, pp. 679–720 (cit. on p. 22).

[Jel05] Zbigniew Jelonek. "On the effective Nullstellensatz". In: *Invent. Math.* 162.1 (2005), pp. 1–17. DOI: 10.1007/s00222-004-0434-8 (cit. on p. 1).

[Jou91] Jean-Pierre Jouanolou. "Le formalisme du résultant". In: *Adv. Math.* 90.2 (1991), pp. 117–263. ISSN: 0001-8708,1090-2082. DOI: 10.1016/0001-8708(91)90031-2 (cit. on pp. 22, 23, 35).

[JS07] Gabriela Jeronimo and Juan Sabia. "Computing multihomogeneous resultants using straight-line programs". In: *J. Symbolic Comput.* 42.1-2 (2007), pp. 218–235. ISSN: 0747-7171,1095-855X. DOI: 10.1016/j.jsc.2006.03.006 (cit. on pp. 23, 32).

[Koi96] Pascal Koiran. "Hilbert's Nullstellensatz is in the polynomial hierarchy". In: *Journal of complexity* 12.4 (1996), pp. 273–286. DOI: https://doi.org/10.1006/jcom.1996.0019 (cit. on p. 2).

[Koi97] Pascal Koiran. "Randomized and deterministic algorithms for the dimension of algebraic varieties". In: *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1997)*. IEEE Computer Society, 1997, pp. 36–45. ISBN: 0818681977. DOI: 10.1109/SFCS.1997.646091 (cit. on pp. 2, 55).

[Kol88] János Kollár. "Sharp effective Nullstellensatz". In: *J. Amer. Math. Soc.* 1.4 (1988), pp. 963–975. DOI: 10.2307/1990996 (cit. on p. 1).

[KP96] T. Krick and L. M. Pardo. "A computational method for diophantine approximation". In: *Algorithms in Algebraic Geometry and Applications*. Ed. by Laureano González-Vega and Tomás Recio. Basel: Birkhäuser Basel, 1996, pp. 193–253. ISBN: 978-3-0348-9104-2 (cit. on p. 27).

[KPS01] Teresa Krick, Luis Miguel Pardo, and Martín Sombra. "Sharp estimates for the arithmetic Nullstellensatz". In: *Duke Mathematical Journal* 109.3 (2001), pp. 521–598. DOI: 10.1215/S0012-7094-01-10934-4 (cit. on p. 1).

[LST21] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. "Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits". In: *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*. 2021, pp. 804–814. DOI: 10.1109/FOCS52979.2021.00083 (cit. on p. 23).

[LST25] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. "Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits". In: *J. ACM* 72.4 (2025), 26:1–26:35. DOI: 10.1145/3734215 (cit. on p. 5).

[Mat70] Ju. V. Matijasevič. "The Diophantineness of enumerable sets". In: *Dokl. Akad. Nauk SSSR* 191 (1970), pp. 279–282. ISSN: 0002-3264 (cit. on p. 1).

[MM82] Ernst W. Mayr and Albert R. Meyer. "The complexity of the word problems for commutative semigroups and polynomial ideals". In: *Adv. in Math.* 46.3 (1982), pp. 305–329. ISSN: 0001-8708. DOI: 10.1016/0001-8708(82)90048-2 (cit. on p. 4).

[MS58] Nathaniel Macon and Abraham Spitzbart. "Inverses of Vandermonde matrices". In: *The American Mathematical Monthly* 65.2 (1958), pp. 95–100. DOI: 10.2307/2308881 (cit. on p. 16).

[MSW95] Alexander P. Morgan, Andrew J. Sommese, and Charles W. Wampler. "A Product-Decomposition Bound for Bezout Numbers". In: *SIAM Journal on Numerical Analysis* 32.4 (1995), pp. 1308–1325. DOI: 10.1137/0732061 (cit. on p. 32).

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. "Derandomizing Arthur-Merlin Games using Hitting Sets". In: *Comput. Complex.* 14.3 (2005), pp. 256–279. DOI: 10.1007/S00037-005-0197-7 (cit. on p. 2).

[RT92] John H. Reif and Stephen R. Tate. "On threshold circuits and polynomial computation". In: *SIAM J. Comput.* 21.5 (1992), pp. 896–908. ISSN: 0097-5397. DOI: 10.1137/0221053 (cit. on p. 14).

[Sei74] A. Seidenberg. "Constructions in algebra". In: *Trans. Amer. Math. Soc.* 197 (1974), pp. 273–313. ISSN: 0002-9947,1088-6850. DOI: 10.2307/1996938 (cit. on p. 1).

[Som99] Martín Sombra. "A sparse effective Nullstellensatz". In: *Adv. in Appl. Math.* 22.2 (1999), pp. 271–295. DOI: 10.1006/aama.1998.0633 (cit. on p. 1).

[Tod91] Seinosuke Toda. "PP is as Hard as the Polynomial-Time Hierarchy". In: *SIAM Journal on Computing* 20.5 (1991), pp. 865–877. DOI: 10.1137/0220053 (cit. on p. 3).

[Tor91] Jacobo Torán. "Complexity Classes Defined by Counting Quantifiers". In: *J. ACM* 38.3 (1991), pp. 753–774. DOI: 10.1145/116825.116858 (cit. on p. 3).

[Vol99] Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999. DOI: doi.org/10.1007/978-3-662-03927-4 (cit. on p. 10).

[Wag86] Klaus W. Wagner. "The Complexity of Combinatorial Problems with Succinct Input Representation". In: *Acta Informatica* 23.3 (1986), pp. 325–356. DOI: 10.1007/BF00289117 (cit. on p. 3).

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il