Electronic Colloquium on Computational Complexity, Report No. 63 (2025)

Algebraic Pseudorandomness in VNC^0

Robert Andrews*

May 15, 2025

Abstract

We study the arithmetic complexity of hitting set generators, which are pseudorandom objects used for derandomization of the polynomial identity testing problem. We give new explicit constructions of hitting set generators whose outputs are computable in VNC^0, i.e., can be computed by arithmetic formulas of constant size. Unconditionally, we construct a VNC^0-computable generator that hits arithmetic circuits of constant depth and polynomial size. We also give conditional constructions, under strong but plausible hardness assumptions, of VNC^0-computable generators that hit arithmetic formulas and arithmetic branching programs of polynomial size, respectively. As a corollary of our constructions, we derive lower bounds for subsystems of the Geometric Ideal Proof System of Grochow and Pitassi.

Constructions of such generators are implicit in prior work of Kayal on lower bounds for the degree of annihilating polynomials. Our main contribution is a construction whose correctness relies on circuit complexity lower bounds rather than degree lower bounds.

1 Introduction

1.1 Polynomial Identity Testing

Algebraic complexity is a vibrant subarea of complexity theory that studies computation of polynomial and rational functions using basic arithmetic operations. Like boolean complexity theory, algebraic complexity enjoys a rich theory of pseudorandomness, with the polynomial identity testing (PIT) problem playing a central role. The input to the PIT problem is an arithmetic circuit, and the goal is to decide whether the circuit computes the identically zero polynomial. This problem can be efficiently solved with randomness using the Schwartz–Zippel lemma [Sch80; Zip79], which says that if a degree-d polynomial f is nonzero, then with probability at least $1/2$ a randomly-chosen point from a grid of side-length $2d$ will lead to a nonzero evaluation of f. A great deal of work has gone into designing efficient deterministic algorithms for polynomial identity testing, leading to beautiful constructions and connections to other areas of computer science and mathematics.

Algorithms for PIT are often designed by constructing *hitting set generators*, which play a role analogous to pseudorandom generators in boolean complexity. For a complexity class $\mathscr{C}$, a hitting set generator $\mathcal{G}$ for $\mathscr{C}$ is a (family of) polynomial map(s) $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ such that for every polynomial $f \in \mathscr{C}$, we have $f = 0$ if and only if $f \circ \mathcal{G} = 0$.[1] Conceptually, one can think of a generator as

*Cheriton School of Computer Science, University of Waterloo. Part of this work was done at the Institute for Advanced Study and was supported by NSF grant CCF-1900460 and the Erik Ellentuck Endowed Fellowship Fund. Email: randrews@uwaterloo.ca.

[1] Formally, we consider *families* of polynomials $(f_1, f_2, \ldots)$ and families of polynomial maps $(\mathcal{G}_1, \mathcal{G}_2, \ldots)$, rather than a single polynomial f and a single map $\mathcal{G}$. We will gloss over this distinction throughout the introduction for the sake of readability, focusing instead on single elements f_n and $\mathcal{G}_n$ of the respective families.

ISSN 1433-8092

mapping ℓ truly random field elements to n pseudorandom field elements. If the generator $\mathcal{G}$ can be implemented by an efficient algorithm, then we are led to an improved deterministic PIT algorithm for $\mathscr{C}$-circuits: given a circuit C, test the composition $C \circ \mathcal{G}$ by brute force. The running time of the naïve brute-force algorithm for PIT has an exponential dependence on the number of variables, and the generator $\mathcal{G}$ improves this exponent from n to the smaller ℓ, a parameter referred to as the *seed length* of the generator. The usual aim is to construct generators with seed length that is as small as possible, since this is the most important parameter for improving the running time of the deterministic algorithm.

Numerous constructions of hitting set generators are known, both conditional and unconditional. In this work, we will be interested in designing hitting set generators for strong circuit classes, at or beyond the frontier of our ability to prove super-polynomial lower bounds for explicit polynomials.[2] Starting with Kabanets and Impagliazzo [KI04], who adapted the Nisan–Wigderson [NW94] generator to the algebraic setting, there has been a successful line of work constructing hitting set generators for strong circuit classes—including general, unrestricted circuits—under hardness assumptions [DSY09; CKS19; And20; ST21b; GKSS22]. In fact, some of the lower bounds assumed by these works have since been proven unconditionally, leading to new deterministic algorithms for PIT! Specifically, the super-polynomial lower bounds of Limaye, Srinivasan, and Tavenas [LST21] against constant-depth arithmetic circuits, combined with the hardness-to-randomness theorem of Chou, Kumar, and Solomon [CKS19], led to the first deterministic sub-exponential time algorithm to test polynomial identities written as constant-depth circuits.

Recently, Chatterjee and Tengse [CT25] raised a very interesting question about the complexity of hitting set generators. They asked if it is possible to construct a hitting set generator that is computable in some small complexity class $\mathscr{C}$, yet appears pseudorandom to a larger complexity class $\mathscr{D} \supsetneq \mathscr{C}$. This sort of parameter regime is common throughout cryptography, so they termed such a generator a *cryptographic* hitting set generator. The question of constructing cryptographic hitting set generators is an extremely interesting one, and is the focus of our work. Aside from this question's inherent interest, one might expect the techniques underlying the construction of a cryptographic hitting set generator to have other applications within algebraic complexity theory. For example, the analogous question of constructing low-complexity pseudorandom generators was answered in a beautiful work of Applebaum, Ishai, and Kushilevitz [AIK06], and the techniques therein have found numerous applications within cryptography and complexity, such as to the recent study of the range avoidance problem [RSW22].

Most known hardness-randomness connections in algebraic complexity cannot hope to produce a hitting set generator that operates in this cryptographic regime of parameters. Almost all generators that come from algebraic hardness-randomness are *reconstructive*, meaning that their correctness proofs follow a common template. In a reconstructive proof of correctness, the generator $\mathcal{G}$ is proven correct by an argument of the following form: suppose some explicit polynomial f, such as the $n \times n$ permanent, is hard to compute. Given a nonzero circuit C of size s that satisfies $C \circ \mathcal{G} = 0$, there is a reconstruction procedure that modifies C into a circuit C' of size $s + t$ that computes the supposedly-hard polynomial f. If $s + t \ll \mathrm{size}(f)$, where $\mathrm{size}(f)$ is the complexity of f, then we arrive at a contradiction. The upshot of this argument is that if the circuit C is sufficiently small—in particular, when the circuit C is of size $s \ll \mathrm{size}(f)$—the hardness of f implies that the composition $C \circ \mathcal{G}$ must be nonzero, i.e., that $\mathcal{G}$ hits small circuits. The bottleneck in this argument is that the generator $\mathcal{G}$ is usually defined using the hard polynomial f (for example, $\mathcal{G}$ may be the Nisan–Wigderson generator applied to f), so the complexity of $\mathcal{G}$ is necessarily bounded from below

[2] For results on polynomial identity testing for weaker circuit classes, we refer the reader to the recent survey of Dutta and Ghosh [DG24], as well as the surveys of Saxena [Sax09; Sax14].

by size(f). Because the reconstruction argument only proves that $\mathcal{G}$ is pseudorandom against circuits of size $s \ll \text{size}(f) \leqslant \text{size}(\mathcal{G})$, this proof template is doomed to fail in constructing a cryptographic hitting set generator. To design cryptographic hitting set generators for strong circuit classes, we need to avoid this reconstructive approach.

One instance where the overhead from reconstruction can be avoided is found in the work of Andrews and Forbes [AF22]. They gave a subexponential-time algorithm to test polynomial identities that are written as constant-depth circuits. The same algorithmic result was already obtained by the previously-mentioned work of Limaye, Srinivasan, and Tavenas [LST21]. However, these two algorithms differ in the complexity of the hitting set generator underlying the algorithm: the generator of [LST21] is reconstructive, whereas the generator of [AF22] provably has smaller complexity than the circuits it hits. Unlike most works in algebraic hardness-randomness, Andrews and Forbes [AF22] do not design a generator that is based on evaluations of hard functions, but rather design the generator so that its annihilator ideal consists only of hard polynomials. For a generator $\mathcal{G}$, its *annihilator ideal*, denoted by $\text{Ann}(\mathcal{G})$, is the set of all polynomials f such that $f \circ \mathcal{G} = 0$. In principle, one could show that a generator hits a circuit class $\mathscr{C}$ by showing that all nonzero polynomials in the annihilator ideal are so complex that they lie outside the circuit class $\mathscr{C}$. Proving a statement like this is necessary for the proof of correctness of a generator, but these statements are usually not the core thrust of the argument and only arise as a byproduct of the correctness proof. As we will see, adopting this viewpoint will be useful for constructing further examples of generators that hit circuits more complex than the generator itself.

1.2 Cryptographic Generators from Degree Bounds for Annihilating Polynomials

Although the question of constructing cryptographic hitting set generators in algebraic complexity is a recent one, prior work on degree bounds for annihilating polynomials leads, at least implicitly, to constructions of cryptographic generators. Kayal [Kay09, Theorem 12] showed that over a field of characteristic zero, any annihilator of the $n+1$ polynomials

$$\begin{aligned} g_1(\overline{x}) &= x_1^d - 1 \\ &\vdots \\ g_n(\overline{x}) &= x_n^d - 1 \\ g_{n+1}(\overline{x}) &= x_1 + x_2 + \cdots + x_n - n \end{aligned}$$

must have degree at least d^n. Equivalently, the corresponding generator $\mathcal{G} : \mathbb{F}^n \to \mathbb{F}^{n+1}$ given by $\mathcal{G}(\overline{x}) = (g_1(\overline{x}), \ldots, g_{n+1}(\overline{x}))$ hits all polynomials of degree less than d^n. Because many restricted forms of arithmetic circuits, such as arithmetic formulas and branching programs, necessarily compute low-degree polynomials, this generator hits powerful classes of circuits. For example, size-s arithmetic branching programs cannot compute polynomials of degree greater than s, so this generator hits *all* branching programs of size less than d^n. This is a bit unusual. The best-known lower bounds against arithmetic branching programs are only quadratic [CKSV22], so intuition from the hardness versus randomness paradigm suggests we should only expect to have constructions of generators that hit branching programs of size up to $O(n^2)$.

The preceding example of Kayal [Kay09] fits into the cryptographic regime, as the outputs are extremely simple to compute. Taking $d = 2$, we obtain a generator where the first n outputs can be computed by formulas of constant size, the last output can be computed by a formula of size n and depth two, and yet the generator hits all branching programs of size less than 2^n. Although this generator only stretches its input by one field element in length, it is possible to improve the stretch

of the generator by invoking $n^{1-\varepsilon}$ copies in parallel, each on a fresh set of n^{ε} variables. This results in a generator $\mathcal{G}' : \mathbb{F}^n \to \mathbb{F}^{n+n^{1-\varepsilon}}$ that stretches its input by $n^{1-\varepsilon}$ field elements. Kayal's construction can also be modified to obtain a generator with similar parameters where *all* outputs are computable by constant-size formulas. To do this, we encode the final output g_{n+1} with additional variables $y_2, \ldots, y_{n-1}$ via

$$\begin{aligned} h_{n+1}(\overline{x}, \overline{y}) &= x_1 + x_2 - y_2 \\ h_{n+2}(\overline{x}, \overline{y}) &= y_2 + x_3 - y_3 \\ &\vdots \\ h_{2n-1}(\overline{x}, \overline{y}) &= y_{n-1} + x_n - n. \end{aligned}$$

Kayal's lower bound on the degree of the annihilator extends to this variation on his example, so we already have constructions of generators where each output is computable by a formula of constant size, yet the generator itself hits polynomials of much higher complexity.

1.3 The Ideal Proof System

Not only is the problem of constructing cryptographic hitting set generators interesting in its own right, but it is also closely tied to open problems in algebraic proof complexity. The central goal of propositional proof complexity is to understand the NP versus coNP problem via the complexity of the coNP-complete unsatisfiability problem: given a boolean formula φ, accept φ if and only if φ is unsatisfiable. The typical setting is to first fix a proof system, and then try to find families of boolean formulas $(\varphi_n)_{n\in\mathbb{N}}$ such that any proof π_n of the unsatisfiability of φ_n requires length that is super-polynomial in n. Numerous proof systems based on different areas of mathematics, including logic, algebra, and geometry, have been considered.

Most relevant to our work are proof systems based on algebraic reasoning. Without loss of generality, we may assume that our unsatisfiable boolean formula φ is in 3CNF form. If φ is an n-variate 3CNF formula with m clauses, there is a translation of φ to a system $\mathcal{F}$ of $n+m$ polynomial equations such that $\mathcal{F}$ is satisfiable if and only if φ is satisfiable. This system $\mathcal{F}$ consists of the n boolean axioms $x_i^2 - x_i = 0$, which force the variables x_i to be $\{0,1\}$-valued, and for each clause of the form $(x_1 \oplus b_1) \vee (x_2 \oplus b_2) \vee (x_3 \oplus b_3)$, the trivariate equation

$$(x_1 - (1 - b_1))\,(x_2 - (1 - b_2))\,(x_3 - (1 - b_3)) = 0.$$

It is easy to see that any boolean assignment to the x variables satisfies this equation if and only if the same assignment satisfies the corresponding clause $(x_1 \oplus b_1) \vee (x_2 \oplus b_2) \vee (x_3 \oplus b_3)$. Thus, if we want to prove that φ is unsatisfiable, we can instead try to prove the unsatisfiability of the resulting system of polynomial equations $\mathcal{F}$.

How does one prove that a system of polynomial equations is unsatisfiable? The answer comes from Hilbert's Nullstellensatz. Over an algebraically closed field $\mathbb{F}$, the system of equations

$$f_1(\overline{x}) = \cdots = f_m(\overline{x}) = 0$$

is unsatisfiable if and only if there are polynomials $g_1, \ldots, g_m \in \mathbb{F}[x_1, \ldots, x_n]$ such that

$$\sum_{i=1}^{m} f_i(\overline{x})\, g_i(\overline{x}) = 1.$$

We can take the polynomials $\{g_1, \ldots, g_m\}$ to be our proof of unsatisfiability. Our goal, then, is to understand the complexity of the simplest refutation $\{g_1, \ldots, g_m\}$ of the system of equations $\mathcal{F}$.

The complexity of the proof depends on the choice of proof system. The Nullstellensatz proof system [BIKPP96; BIK+96] measures the length of the proof by the total number of monomials in the g_i. A slightly stronger system is the Polynomial Calculus [Raz98; IPS99], where we are allowed to write the derivation of 1 from the f_i in a line-by-line manner, but we still pay for the number of monomials written on every line. Much stronger is the Ideal Proof System (IPS), introduced by Grochow and Pitassi [GP18], which allows us to write the g_i succinctly as arithmetic circuits. Given that refutations in IPS are written as arithmetic circuits, one would hope that techniques for proving arithmetic circuit lower bounds will eventually lead to lower bounds for the IPS.

This hope has played out successfully in recent years. Forbes, Shpilka, Tzameret, and Wigderson [FSTW21] introduced two techniques to infer IPS lower bounds from arithmetic circuit lower bounds, and applied these techniques for restricted circuit classes to conclude unconditional lower bounds. The first technique, known as the functional lower bound method, has been further refined and studied by Govindasamy, Hakoniemi, and Tzameret [GHT22] and Hakoniemi, Limaye, and Tzameret [HLT24]. Here, the hard system of equations $\mathcal{F}$ typically consists of boolean axioms together with a sparse polynomial that corresponds to an instance of subset sum, possibly lifted with an appropriate gadget.[3] This method is capable of proving strong lower bounds; for example, Hakoniemi, Limaye, and Tzameret [HLT24] proved super-polynomial lower bounds on the size of constant-depth IPS refutations for such systems of equations. However, current applications of the functional lower bound method are only able to prove lower bounds against refutations of individual degree $O(\log \log n)$. This is to be expected, as functional lower bounds without the individual degree constraint, even for depth-four arithmetic circuits, would lead to new lower bounds in boolean complexity theory [FKS16].

The second technique introduced by Forbes, Shpilka, Tzameret, and Wigderson [FSTW21] is based on hardness of multiples. This method is also capable of proving strong lower bounds: later work by Andrews and Forbes [AF22] proved super-polynomial lower bounds on the size of constant-depth IPS refutations for a system of equations related to matrix rank. Unlike the functional lower bound method, the lower bounds produced from hardness of multiples do not require the IPS refutation to be of small individual degree. However, the drawback to this method is that in order to prove a lower bound against IPS certificates computed by $\mathscr{C}$-circuits, there must be at least one polynomial f in the hard instance $\mathcal{F}$ which itself cannot be computed efficiently by $\mathscr{C}$-circuits. This limitation is inherent to the method, as the lower bound against IPS is derived from circuit lower bounds for one (or more) of the polynomials in the hard instance $\mathcal{F}$.

Strong conditional lower bounds for IPS are also known. Alekseev, Grigoriev, Hirsch, and Tzameret [AGHT24] proved super-polynomial lower bounds on the size of constant-free IPS refutations of the binary value principle, a system of equations that asserts a natural number n given in binary is negative. Their lower bound assumes the τ-conjecture of Shub and Smale, which asserts that the straight-line program complexity of computing (a multiple of) the integer $n!$ is bounded from below by $\log^{\omega(1)} n$. Santhanam and Tzameret [ST21a] showed that under the assumption VP $\neq$ VNP, there is a sequence of 3CNF formulas that require IPS refutations of super-polynomial size. One drawback of this result is that the sequence of CNF formulas considered by Santhanam and Tzameret [ST21a] may be satisfiable; if this were the case, then the lower bound becomes trivial, as the soundness of IPS implies that it cannot refute a system of satisfiable equations.

Despite the success so far in bringing techniques from arithmetic circuit complexity to bear on IPS, we still seem far from proving unconditional lower bounds for systems of polynomial equations that encode unsatisfiable 3CNF formulas, even against weak subsystems of IPS. Current applications

[3] These hard instances are closely related to the previously-discussed example of Kayal [Kay09] that leads to lower bounds on the degree of annihilating polynomials.

of the functional lower bound method are limited to proving lower bounds against refutations of low individual degree, and some formulations of the method provably cannot lead to lower bounds for boolean instances [HLT24]. The method of lower bounds for multiples can prove unconditional lower bounds against fragments of IPS with no restrictions on the individual degree of the refutation, but the method requires the hard instance to contain polynomials of large circuit complexity, which precludes its application to systems of equations that encode a 3CNF formula. As a step towards proving IPS lower bounds for such systems, can we prove IPS lower bounds for any system of equations where each polynomial in the system can be described by an arithmetic formula of constant size?

1.4 Hard Families of Simple Polynomials from Nullstellensatz Degree Bounds

Just as prior work on annihilating polynomials led to simple constructions of cryptographic hitting set generators, existing work on lower bounds for Nullstellensatz degree—in particular, examples demonstrating the tightness of these bounds—easily leads to families of simple polynomials that are hard for fragments of IPS. Recall that the Nullstellensatz says that if $f_1 = \cdots = f_m = 0$ is an unsatisfiable system of equations, then there are polynomials $g_1, \ldots, g_m$ such that $\sum_{i=1}^{m} f_i g_i = 1$. Of particular relevance to our work are degree bounds for the Nullstellensatz: given the polynomials f_i, what is the smallest possible degree of the polynomials g_i that witness the identity $\sum_{i=1}^{n} f_i g_i = 1$? A long line of work [Her26; Bro87; CGH88; Kol88; FG90; Som99; KPS01; Jel05] has established various bounds on the degrees and heights of such polynomials. For example, we know that such polynomials g_i can always be found with degree $\deg(g_i) \leqslant d^n$, where $d = \max_i \deg(f_i)$ is the maximum degree of the f_i. An example due to Masser and Philippon (see [Bro87]) shows that bounds of this shape are tight. In particular, if we take

$$\begin{aligned} f_1(\overline{x}) &= x_1^d \\ f_2(\overline{x}) &= x_1 - x_2^d \\ &\vdots \\ f_{n-1}(\overline{x}) &= x_{n-2} - x_{n-1}^d \\ f_n(\overline{x}) &= 1 - x_{n-1} x_n^{d-1}, \end{aligned}$$

then in any expression $\sum_{i=1}^{n} f_i g_i = 1$, the polynomial g_1 must satisfy $\deg(g_1) \geqslant d^n - d^{n-1}$. This degree lower bound implies similar lower bounds on the size of IPS refutations of the above system when the refutation is written as an arithmetic formula or arithmetic branching program. The lower bound follows from the fact that an arithmetic formula or branching program of size s can only compute polynomials of degree at most s, so no formula or branching program of size less than $d^n - d^{n-1}$ can compute a refutation of this system.

At first glance, it appears that we have made progress towards IPS lower bounds for refuting 3CNF formulas. The example of Masser and Philippon above gives us a system of polynomial equations, each of which can be encoded by a constant-size arithmetic formula, that requires large refutations in powerful subsystems of IPS. Unfortunately, this line of reasoning cannot lead to IPS lower bounds for systems of polynomials that encode 3CNF formulas. An unsatisfiable 3CNF formula on n variables always admits a degree-$O(n)$ refutation [GP18], so degree bounds will not result in new lower bounds on the complexity of refuting 3CNF formulas. To make progress towards proving lower bounds for IPS refutations of 3CNF formulas, we need techniques rooted in finer complexity measures than degree.

1.5 Our Contributions

We now describe our results. As we have seen in Sections 1.2 and 1.4, prior work on degree bounds can be used to construct cryptographic hitting set generators and hard instances for the Ideal Proof System. The main contribution of our work is a new construction of cryptographic hitting set generators whose correctness is based on arithmetic circuit lower bounds, not degree lower bounds. As a corollary, we will obtain families of simple polynomials that are hard to refute in subsystems of the the Geometric Ideal Proof System, itself a restricted form of the Ideal Proof System [GP18, Appendix B]. Although the theorem statements below already follow from prior work, the constructions appearing in our proofs do not, and we believe there is value in developing techniques in algebraic pseudorandomness and proof complexity that go beyond degree lower bounds.

Our first result is an explicit construction of a low-complexity hitting set generator that hits VAC^0, the class of polynomials computed by constant-depth circuits of polynomial size. Each output of our generator is computable in VNC^0, i.e., can be computed by an arithmetic formula of constant size.

Theorem 1.1 (see Theorem 4.3)**.** *Let $\mathbb{F}$ be a field of characteristic zero.*[4] *There is a VNC^0-computable hitting set generator with stretch $n^{0.99}$ that hits VAC^0.*

Under strong but reasonable hardness assumptions, the same techniques yield VNC^0-computable hitting set generators that hit larger circuit classes. Our first conditional construction yields a VNC^0-computable generator that hits VF, which corresponds to families of polynomials that are computable by polynomial-size formulas. The correctness of this construction relies on the assumption that the border formula complexity of the determinant is super-polynomial.

Theorem 1.2 (see Theorem 4.6)**.** *Let $\mathbb{F}$ be a field of characteristic zero. If the border formula complexity of the $n \times n$ determinant is $n^{\omega(1)}$, then there is a VNC^0-computable hitting set generator that hits VF.*

Although it is commonly conjectured that the formula complexity of the determinant is $n^{\omega(1)}$, it is less clear whether we should expect the same lower bound to hold for border formulas, as border computation is poorly understood even in very weak settings. Despite this gap in our understanding, we find it conceivable that the determinant does require border formulas of super-polynomial size. The reason for this is that recent progress on arithmetic circuit lower bounds [LST21; TLS22; KS22; KS23; FLST24; For24] has relied on the use of complexity measures based on matrix rank. Because matrix rank is lower semi-continuous, these lower bounds often directly imply lower bounds on border complexity.[5] It is unclear if these methods will prove lower bounds against arithmetic formulas in the near future, and if they do, the hard polynomial may not be the determinant. However, in light of the recent success of rank-based methods in proving lower bounds, coupled with the fact that many of these lower bounds apply to the determinant, it is not out of the question that when we manage to prove lower bounds against arithmetic formulas, the methods used will be rank-based and will apply to the determinant. In that case, we will have corresponding lower bounds on the border formula complexity of the determinant.

Our second conditional construction produces a VNC^0-computable generator that hits polynomials that are computable by polynomial-size arithmetic branching programs, a class commonly known as VBP. This result assumes there is a family of polynomials that can be computed by polynomial-size arithmetic circuits but not by polynomial-size arithmetic branching programs.

[4] For technical reasons, we can only prove that our generator is pseudorandom over fields of characteristic zero or sufficiently large characteristic. See Section 4 for an explanation of the underlying issue.

[5] See e.g. [AF22, Section 6.1] for an explanation of how the lower bound of Limaye, Srinivasan, and Tavenas [LST21] implies a lower bound on border complexity.

Theorem 1.3 (see Theorem 4.9)**.** *Let $\mathbb{F}$ be a field of characteristic zero. If there is a family of polynomials in* VP *that require arithmetic branching programs of super-polynomial size, then there is a* VNC^0*-computable hitting set generator that hits* VBP*.*

As a corollary of our generator constructions, we obtain new lower bounds for subsystems of the Geometric Ideal Proof System of Grochow and Pitassi [GP18, Appendix B], a restricted form of the Ideal Proof System. An easy observation, already made by Grochow, Kumar, Saks, and Saraf [GKSS17], shows that hitting set generators immediately yield hard instances for the Geometric Ideal Proof System. The complexity of the generator upper bounds the complexity of the equations in the hard instance, and the hitting property of the generator is used to infer the lower bound against Geometric IPS. Because we construct VNC^0-computable generators against VAC^0, we obtain a system of equations where every equation can be written as an arithmetic formula of constant size, but no Geometric IPS refutation can be computed by a circuit of constant depth and polynomial size.

Theorem 1.4 (see Theorem 5.2)**.** *Let $\mathbb{F}$ be a field of characteristic zero. There is an explicit system of polynomial equations $\mathcal{F}_n$ such that (1) each equation in $\mathcal{F}_n$ depends on at most* 3 *variables, (2) the system $\mathcal{F}_n$ can be refuted by Geometric IPS, and (3) any Geometric IPS refutation of $\mathcal{F}_n$ cannot be computed by a circuit of constant depth and polynomial size.*

We also prove conditional lower bounds against Geometric IPS refutations computed by arithmetic formulas (Theorem 5.3) and arithmetic branching programs (Theorem 5.4). The conditions used to prove these lower bounds are the same ones used to construct VNC^0-computable generators that hit arithmetic formulas and arithmetic branching programs, respectively.

Although our hard instances do not correspond to encodings of 3CNF formulas, we view these results as progress towards proving lower bounds for refutations of unsatisfiable boolean formulas. As mentioned in Section 1.4, unsatisfiable 3CNF formulas on n variables can always be refuted in degree $O(n)$, so any method of proving lower bounds on the complexity of refuting 3CNF's must tolerate the existence of low-degree refutations. In all of our lower bounds for Geometric IPS, the hard systems of equations admit refutations of degree $n^{O(1)}$. To the best of our knowledge, this is the first example of a system of equations where each equation can be expressed by an arithmetic formula of constant size, the system admits low-degree refutations, and the system is provably hard to refute in a nontrivial subsystem of IPS.

1.6 Our Techniques

We obtain our VNC^0-computable hitting set generators through a transformation that takes a known hitting set generator $\mathcal{G}$ and compiles it into a related generator $\mathcal{G}'$ that has much lower complexity, yet retains the hitting property of $\mathcal{G}$. A similar high-level approach was taken by Applebaum, Ishai, and Kushilevitz [AIK06] to obtain NC^0-computable one-way functions and pseudorandom generators in the boolean setting. There are some superficial similarities between our work and theirs, since both works obtain the new generator $\mathcal{G}'$ through an encoding of the computation of $\mathcal{G}$.

However, one difference between our work and [AIK06] is the set of requirements placed on the high-complexity generator $\mathcal{G}$. Applebaum, Ishai, and Kushilevitz [AIK06] require $\mathcal{G}$ to be of sufficiently low complexity (say, NC^1-computable), but are agnostic to the particular choice of the generator $\mathcal{G}$. In contrast, our work does not place a requirement on the complexity of the starting generator, but we are only able to handle generators $\mathcal{G}$ that are of the specific form

$$\mathcal{G} : (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n, f(\overline{x})),$$

where $f(\overline{x})$ is a polynomial.

We require the starting generator to be of this form because it provides sufficient algebraic structure to completely analyze the resulting low-complexity generator $\mathcal{G}'$. To every generator $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$, one can associate its *annihilator ideal* $\mathrm{Ann}(\mathcal{G})$, the set of polynomials $h \in \mathbb{F}[z_1, \ldots, z_n]$ such that $h \circ \mathcal{G} = 0$, i.e., the composition of h and $\mathcal{G}$ results in the identically zero polynomial. To prove that a generator $\mathcal{G}$ hits a circuit class $\mathscr{C}$, it is both necessary and sufficient to show that every nonzero polynomial in the annihilator $\mathrm{Ann}(\mathcal{G})$ cannot be computed within the resource bounds of $\mathscr{C}$.

For some generators, we can completely characterize the ideal $\mathrm{Ann}(\mathcal{G})$, which a useful first step towards proving lower bounds for $\mathrm{Ann}(\mathcal{G})$. In the case where $\mathcal{G}$ is of the form $\mathcal{G}(\overline{x}) = (x_1, \ldots, x_n, f(\overline{x}))$, it is easy to show that the annihilator ideal is given by

$$\mathrm{Ann}(\mathcal{G}) = \langle z_{n+1} - f(z_1, \ldots, z_n) \rangle \subseteq \mathbb{F}[z_1, \ldots, z_{n+1}].$$

That is, every annihilator of $\mathcal{G}$ is a multiple of the polynomial $z_{n+1} - f(z_1, \ldots, z_n)$. Because this ideal is generated by a single polynomial (i.e., is a principal ideal), we can leverage existing techniques on polynomial factorization to prove lower bounds for $\mathrm{Ann}(\mathcal{G})$. The argument proceeds by contradiction: if there is a small $\mathscr{C}$-circuit that computes an element of $\mathrm{Ann}(\mathcal{G})$, and if $\mathscr{C}$-circuits are polynomially-closed under factorization, then there is a small $\mathscr{C}$-circuit for $z_{n+1} - f(z_1, \ldots, z_n)$. Thus, if the circuit class $\mathscr{C}$ is closed under factorization (as are VP [Kal87] and VBP [ST21b]), then to prove lower bounds for $\mathrm{Ann}(\mathcal{G})$, it suffices to prove a lower bound on $f(z_1, \ldots, z_n)$. This is a much easier task, since we only have to reason about a single polynomial f and not an arbitrary multiple of f. In some cases, the circuit class $\mathscr{C}$ is not known to be polynomially-closed under factorization. Despite this, we can still make use of results that show for a *specific choice* of f, lower bounds on the $\mathscr{C}$-circuit complexity of f imply lower bounds for all multiples of f.

So far, we have seen that it is possible to completely understand generators of the form $\mathcal{G}(x_1, \ldots, x_n) = (x_1, \ldots, x_n, f(\overline{x}))$. On its own, this generator has no hope of producing a generator that is of lower complexity than the circuits it hits, since there is always an annihilator of complexity comparable to f.

To obtain a generator of lower complexity, we replace the single output $f(\overline{x})$ by a sequence of $s+1$ outputs that encode a size-s circuit Φ that computes $f(\overline{x})$.[6] For each internal gate of Φ, we introduce a fresh variable y_i, and we include the polynomial

$$y_i - y_j \odot y_k$$

in the output of the generator, where $\odot \in \{+, \times\}$ is the operation labeling the i^{th} gate and the children of the i^{th} gate are gates j and k.[7] We also include y_s as an output so that the generator stretches its $n+s$ inputs to $n+s+1$ outputs. It is clear from the definition that each output of this new generator $\mathcal{G}'$ can be computed by an arithmetic formula of constant size, but it is not obvious why $\mathcal{G}'$ preserves any pseudorandom properties the original generator $\mathcal{G}$ had.

To show that $\mathcal{G}'$ is pseudorandom, we follow the approach suggested above: we determine the annihilator ideal $\mathrm{Ann}(\mathcal{G}')$ and subsequently prove lower bounds on the complexity of all nonzero polynomials in $\mathrm{Ann}(\mathcal{G}')$. As we saw, the annihilator ideal $\mathrm{Ann}(\mathcal{G})$ of the starting generator was generated by $z_{n+1} - f(z_1, \ldots, z_n)$, so we could infer lower bounds on $\mathrm{Ann}(\mathcal{G})$ from lower bounds for f and its multiples. For $\mathcal{G}'$, the annihilator ideal $\mathrm{Ann}(\mathcal{G}')$ is again principal and is generated by a polynomial of the form

$$h(\overline{z}) = z_{n+s+1} - f(z_1, \ldots, z_n) + g(\overline{z}),$$

[6] This transformation is similar to the reduction of Circuit-SAT to 3CNF-SAT and was recently used by Grochow [Gro23] to study the PIT instances that arise from verification of IPS refutations.

[7] If the children of the i^{th} gate are not internal gates, we use a different polynomial, but this is a technical point that does not meaningfully impact the overview here. See Definition 3.1 for the precise definition.

where g is a structured polynomial that should be thought of as an error term for the purposes of this overview. To prove that $\mathcal{G}'$ is pseudorandom, it suffices to prove lower bounds on the complexity of multiples of h. Because h is close to f, one could hope that lower bounds for multiples of f imply comparable lower bounds for multiples of h. This is precisely the route we follow to show that $\mathcal{G}'$ is pseudorandom. By instantiating the construction of $\mathcal{G}'$ with a polynomial f whose multiples are hard for a circuit class $\mathscr{C}$ (possibly under hardness assumptions), we obtain a generator that is computable in VNC^0 yet hits the much larger class $\mathscr{C}$.

We encourage readers who want to see a concrete example of this new generator $\mathcal{G}'$ to skip ahead to Example 3.3. There, we describe and analyze the generator $\mathcal{G}'$ when the polynomial f is taken to be $f(x_1, x_2) = x_1^2 - x_2^2$. This polynomial is too simple to yield a generator with useful pseudorandom properties, but its simplicity allows us to explicitly write down the resulting generator and its annihilator ideal.

1.7 Organization

The rest of this paper is organized as follows. We review preliminary material in Section 2. We then begin in Section 3, where we describe a general construction of VNC^0-computable hitting set generators. Section 4 gives three concrete instantiations of this generator construction. Section 5 uses our generators to obtain lower bounds for the Geometric Ideal Proof System. Finally, we conclude in Section 6 with some open questions.

2 Preliminaries

For a natural number $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, 2, \ldots, n\}$. We abbreviate a vector of variables $(x_1, \ldots, x_n)$ as $\overline{x}$. For a field $\mathbb{F}$, we write $\mathbb{F}[\overline{x}]$ and $\mathbb{F}(\overline{x})$ for the polynomial ring and field of rational functions, respectively, over $\mathbb{F}$ in the variables $x_1, \ldots, x_n$. A *polynomial map* $\mathcal{G}$ is a tuple of polynomials $(g_1(\overline{x}), \ldots, g_n(\overline{x})) \in \mathbb{F}[x_1, \ldots, x_\ell]^n$, which defines a map $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ via $\mathcal{G}(\overline{\alpha}) = (g_1(\overline{\alpha}), \ldots, g_n(\overline{\alpha}))$. We often abuse notation and refer to the induced map $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ as a polynomial map; the two objects are equivalent over infinite fields, but when $\mathbb{F}$ is finite, different tuples of polynomials may induce the same map $\mathbb{F}^\ell \to \mathbb{F}^n$. For a collection of polynomials $f_1, \ldots, f_m \in \mathbb{F}[\overline{x}]$, we write $\langle f_1, \ldots, f_m \rangle$ for the ideal of $\mathbb{F}[\overline{x}]$ generated by $f_1, \ldots, f_m$.

2.1 Arithmetic Circuit Complexity

This subsection recalls basic notions of arithmetic circuit complexity. For more on this topic, we refer the reader to the surveys of Shpilka and Yehudayoff [SY10] and Saptharishi [Sap].

Definition 2.1 (Arithmetic circuits)**.** Let $\mathbb{F}$ be a field and let $\mathbb{F}[\overline{x}]$ be the polynomial ring over $\mathbb{F}$ in n variables. An *arithmetic circuit* Φ is a directed acyclic graph equipped with the following data. The vertices of in-degree zero are called *input gates* and are labeled by either a variable x_i or a constant $\alpha \in \mathbb{F}$. Vertices of positive in-degree are called *internal gates* and are labeled by either addition or multiplication. Each vertex v of Φ computes a polynomial $f_v(\overline{x}) \in \mathbb{F}[\overline{x}]$ in the natural way. If a vertex v has out-degree zero, we call v an *output gate* and say that the circuit Φ computes $f_v(\overline{x})$.

We measure the *size* of Φ by the number of internal gates in the circuit. The *depth* of a circuit is the length of the longest path from any input gate to any output gate. $\lozenge$

We will sometimes insist on working with circuits whose internal gates have fan-in two. A circuit with internal gates of unbounded fan-in can be simulated by a circuit where every internal gate has

fan-in two by replacing each gate of large fan-in with a binary tree of fan-in two gates. This has the potential to increase the size of the circuit from s to $O(s^2)$, which will be negligible for our purposes.

Next, we define arithmetic formulas, which are arithmetic circuits whose the underlying graph is a tree.

Definition 2.2 (Arithmetic formulas)**.** An *arithmetic formula* is an arithmetic circuit whose underlying graph is a tree. Equivalently, an arithmetic formula is an arithmetic circuit in which every gate has out-degree at most one. ◊

We also need the notion of arithmetic branching programs, whose expressive power lies somewhere between that of formulas and circuits.

Definition 2.3 (Arithmetic branching programs)**.** An *arithmetic branching program* is a layered directed acyclic graph $G = (V, E)$ with a single source vertex s and a single sink vertex t. The fact that G is layered means that there is a partition $V = V_0 \sqcup V_1 \sqcup \cdots \sqcup V_d$ such that $V_0 = \{s\}$, $V_d = \{t\}$, and every edge of G is between vertices in V_{i-1} and V_i for some $i \in [d]$. Each edge e of G is labeled by an affine linear polynomial $\ell_e(\overline{x}) \in \mathbb{F}[\overline{x}]$. The branching program computes the polynomial

$$\sum_{P : s \rightsquigarrow t} \prod_{e \in P} \ell_e(\overline{x}),$$

where the sum is over all (s,t)-paths in G. We measure the *size* of the branching program by $|V|$, the total number of vertices. ◊

Having defined our model of computation, we now define the objects we are interested in computing. These are families of polynomials whose degree is polynomially-bounded; such families of polynomials are called p-families.

Definition 2.4 (p-families)**.** Let $\mathbb{F}$ be a field and let $f = (f_n)_{n \in \mathbb{N}}$ be a sequence of polynomials with coefficients in $\mathbb{F}$. We say that f is a *p-family* if $\deg(f_n)$ is polynomially-bounded as a function of n. ◊

We now define the complexity classes we will be interested in throughout this work. Although the definitions of these classes depend on the underlying field $\mathbb{F}$, we suppress this dependence for the sake of readability. The field $\mathbb{F}$ will always be clear from context.

Definition 2.5 (Complexity classes)**.** Let $\mathbb{F}$ be a field.

1. The class VP consists of all p-families $(f_n)_{n \in \mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic circuit of size $n^{O(1)}$.

2. The class VBP consists of all p-families $(f_n)_{n \in \mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic branching program of size $n^{O(1)}$.

3. The class VF consists of all p-families $(f_n)_{n \in \mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic formula of size $n^{O(1)}$.

4. The class VAC^0 consists of all p-families $(f_n)_{n \in \mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic circuit of size $n^{O(1)}$ and depth $O(1)$.

5. The class VNC^0 consists of all p-families $(f_n)_{n \in \mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic circuit of size $O(1)$. ◊

In addition to the standard notion of computing a polynomial via an arithmetic circuit, we will occasionally need to refer to border complexity, which corresponds to a notion of approximate computation using arithmetic circuits.

Definition 2.6 (Border complexity)**.** Let $\mathbb{F}$ be a field an ε be an indeterminate. Let $f(\overline{x}) \in \mathbb{F}[\overline{x}]$ be a polynomial. We say that a circuit Φ over $\mathbb{F}(\varepsilon)$ *approximately computes* $f(\overline{x})$ if Φ computes a polynomial of the form

$$f(\overline{x}) + \varepsilon \cdot g(\overline{x}, \varepsilon),$$

where $g(\overline{x}, \varepsilon) \in \mathbb{F}[\overline{x}, \varepsilon]$ is a polynomial in $x_1, \ldots, x_n$ and ε. We will abbreviate this by saying that Φ computes $f(\overline{x}) + O(\varepsilon)$. If Φ has size s, we say that the *border complexity* of f is bounded by s. $\Diamond$

Over fields of characteristic zero, such as the complex numbers, one should interpret a circuit Φ that approximately computes f as a circuit that computes f in the limit as ε tends towards zero. A potentially more general definition of approximate computation over $\mathbb{C}$ would be to consider a sequence of circuits $(\Phi_m)_{m\in\mathbb{N}}$ that computes a sequence of polynomials $(f_m)_{m\in\mathbb{N}}$ such that $f = \lim_{m\to\infty} f_m$, without requiring a uniform description of the circuits $(\Phi_m)_{m\in\mathbb{N}}$ as in Definition 2.6. However, a result of Alder [Ald84] shows that these two notions of computation coincide over $\mathbb{C}$, so we may work with Definition 2.6 without any loss in generality.

An alternate description of the limit-based definition is that the target polynomial f lies in the Euclidean closure of the set of polynomials of complexity s. By replacing the Euclidean topology with the Zariski topology, one obtains a notion of approximate computation that extends to algebraically closed fields of arbitrary characteristic. As in the preceding paragraph, Alder [Ald84] showed that this notion of approximate computation agrees with Definition 2.6.

Naturally, one can define complexity classes in terms of border complexity.

Definition 2.7 (Border complexity classes)**.** Let $\mathbb{F}$ be a field and let $f = (f_n)_{n\in\mathbb{N}}$ be a p-family. For a complexity class $\mathscr{C}$, we say that $f \in \overline{\mathscr{C}}$ if there is a p-family $g = (g_n)_{n\in\mathbb{N}}$ over $\mathbb{F}(\varepsilon)$ such that $g \in \mathscr{C}$ and for all n, we have $g_n(\overline{x}) = f_n(\overline{x}) + O(\varepsilon)$. $\Diamond$

For example, a p-family $(f_n)_{n\in\mathbb{N}}$ is in $\overline{\mathrm{VP}}$ if there is a sequence of circuits of size $n^{O(1)}$ over $\mathbb{F}(\varepsilon)$ that compute $f_n + O(\varepsilon)$.

2.2 Polynomial Identity Testing

Many deterministic algorithms for polynomial identity testing are obtained by explicitly constructing a hitting set generator. A hitting set generator, defined below, is a pseudorandom object that plays a role analogous to pseudorandom (and hitting set) generators in boolean derandomization, stretching a short seed of truly random field elements into a longer string of pseudorandom field elements.

Definition 2.8 (Hitting set generator)**.** Let $\mathbb{F}$ be a field and let $\mathscr{C}$ and $\mathscr{D}$ be classes of p-families over $\mathbb{F}$. Let $\mathcal{G} = (\mathcal{G}_n : \mathbb{F}^{\ell(n)} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ be a sequence of polynomial maps, where $\mathcal{G}_n = (g_{n,1}, \ldots, g_{n,n}) \in \mathbb{F}[\overline{y}]^n$. We say that $\mathcal{G}$ is a *$\mathscr{D}$-computable hitting set generator for $\mathscr{C}$* if the following conditions hold.

1. Every p-family of the form $g = (g_{n,i_n})_{n\in\mathbb{N}}$ is an element of $\mathscr{D}$.
2. For every p-family $f = (f_n)_{n\in\mathbb{N}} \in \mathscr{C}$ and for all sufficiently large n, we have $(f_n \circ \mathcal{G}_{m(n)})(\overline{y}) = 0$ if and only if $f_n(\overline{x}) = 0$, where $m(n)$ is the number of variables appearing in the polynomial f.

The *seed length* of $\mathcal{G}$ is $\ell(n)$. The *stretch* of $\mathcal{G}$ is $n - \ell(n)$. The *degree* of $\mathcal{G}$ is $d(n) := \max_{i\in[n]} \deg(g_{n,i})$. $\Diamond$

Remark 2.9 (Padding)**.** In Definition 2.8, we adopt the convention that the n^{th} map in the sequence $(\mathcal{G}_n : \mathbb{F}^{\ell(n)} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ has output length n largely as a matter of simplicity. When we construct generators later in Section 4, our constructions will naturally lead to sequences of maps where the n^{th} map has output length $m(n)$ for a polynomially-bounded function $m(n)$. To obtain a generator with output length $m(n)+q < m(n+1)$, we pad the input and output of the generator with $q-m(n)$ fresh variables. This preserves the VNC^0-computability of the generator and does not impact the pseudorandom properties of the generator. ◊

For the sake of readability, we may refer to a single polynomial map $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ as a hitting set generator. This is done with the understanding that there is an underlying sequence of polynomial maps $(\mathcal{G}_n : \mathbb{F}^{\ell(n)} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ and we have implicitly fixed a choice of n.

An explicit construction of a hitting set generator $\mathcal{G}$ for a circuit class $\mathscr{C}$ immediately yields an algorithm to test $\mathscr{C}$-circuits: given a polynomial $f \in \mathscr{C}$, test the composed polynomial $f \circ \mathcal{G}$ by brute force. By the Schwartz–Zippel lemma, testing $f \circ \mathcal{G}$ can be done with $(\deg(f)\deg(\mathcal{G})+1)^\ell$ evaluations. If ℓ is small compared to n and $\deg(\mathcal{G})$ is not large, this yields an improvement over the brute-force identity testing algorithm, which evaluates f at $(\deg(f)+1)^n$ points.

One means of studying the pseudorandom properties of a candidate generator $\mathcal{G}$ is by studying its annihilator ideal $\mathrm{Ann}(\mathcal{G})$, which we now define.

Definition 2.10 (Annihilator ideal)**.** Let $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ be a polynomial map. The *annihilator ideal* of $\mathcal{G}$, denoted by $\mathrm{Ann}(\mathcal{G})$, is the set of all polynomials that vanish when composed with $\mathcal{G}$. Formally,

$$\mathrm{Ann}(\mathcal{G}) \coloneqq \{f \in \mathbb{F}[x_1, \ldots, x_n] : f(g_1(\overline{y}), \ldots, g_n(\overline{y})) = 0\}. \qquad \text{◊}$$

To understand the relation between the pseudorandom properties of a candidate generator $\mathcal{G}$ and its annihilator ideal $\mathrm{Ann}(\mathcal{G})$, suppose we want to show that $\mathcal{G}$ hits some circuit class $\mathscr{C}$. The statement "$\mathcal{G}$ hits $\mathscr{C}$" is equivalent to the containment

$$\mathscr{C} \cap \mathrm{Ann}(\mathcal{G}) \subseteq \{0\}.$$

This containment should be interpreted as saying that every nonzero polynomial in $\mathrm{Ann}(\mathcal{G})$ is too complex to be computed within the resource bounds of the circuit class $\mathscr{C}$. Thus, to prove that a candidate generator $\mathcal{G}$ hits a circuit class $\mathscr{C}$, one can instead prove lower bounds on every nonzero polynomial in $\mathrm{Ann}(\mathcal{G})$. This is the route we will take to prove the correctness of our generator.

We formalize the preceding discussion in the following lemma.

Lemma 2.11. *Let $\mathbb{F}$ be a field and let $\mathscr{C}$ be a class of p-families over $\mathbb{F}$. Let $\mathcal{G} = (\mathcal{G}_n : \mathbb{F}^{\ell(n)} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ be a sequence of polynomials maps. Then $\mathcal{G}$ is a hitting set generator for $\mathscr{C}$ if and only if for every p-family $f = (f_n)_{n\in\mathbb{N}}$ where $f_n \in \mathrm{Ann}(\mathcal{G}_n) \setminus \{0\}$, we have $f \notin \mathscr{C}$.*

2.3 The Ideal Proof System

In this subsection, we recall the Ideal Proof System (IPS) of Grochow and Pitassi [GP18]. This is a proof system that uses algebraic reasoning to prove that systems of polynomial equations do not admit a solution. In propositional proof complexity, the boolean axioms $x_i^2 - x_i = 0$ are often incorporated into the definition of a proof system, as they naturally appear when refuting encodings of unsatisfiable boolean formulas. We will consider the IPS more generally as a proof system for arbitrary systems of polynomial equations, and so we adopt a definition that does not include the boolean axioms by default.

Definition 2.12 (Ideal Proof System)**.** Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be polynomials such that the system of equations $f_1 = \cdots = f_m = 0$ has no solution over $\overline{\mathbb{F}}$, the algebraic closure of $\mathbb{F}$. An *Ideal Proof System (IPS) refutation* of $(f_1, \ldots, f_m)$ is a polynomial $r \in \mathbb{F}[x_1, \ldots, x_n, z_1, \ldots, z_m]$ such that

1. $r(x_1, \ldots, x_n, f_1(\overline{x}), \ldots, f_m(\overline{x})) = 1$, and
2. $r(x_1, \ldots, x_n, 0, \ldots, 0) = 0$. ◊

For our purposes, we will be interested in the Geometric Ideal Proof System [GP18, Appendix B], a natural subsystem of the IPS.

Definition 2.13 (Geometric Ideal Proof System)**.** Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be polynomials such that the system of equations $f_1 = \cdots = f_m = 0$ has no solution over $\overline{\mathbb{F}}$, the algebraic closure of $\mathbb{F}$. A *Geometric Ideal Proof System refutation* of $(f_1, \ldots, f_m)$ is a polynomial $r \in \mathbb{F}[z_1, \ldots, z_m]$ such that

1. $r(f_1(\overline{x}), \ldots, f_m(\overline{x})) = 0$, and
2. $r(0, \ldots, 0) = 1$. ◊

Geometric IPS refutations have a natural interpretation from the viewpoint of algebraic geometry. Consider the polynomial map $\mathbf{f} : \mathbb{F}^n \to \mathbb{F}^m$ given by $(x_1, \ldots, x_n) \mapsto (f_1(\overline{x}), \ldots, f_m(\overline{x}))$. Suppose $r \in \mathbb{F}[z_1, \ldots, z_m]$ is a Geometric IPS refutation of $\mathbf{f}$. The condition $r(f_1(\overline{x}), \ldots, f_m(\overline{x})) = 0$ implies the image of the map $\mathbf{f}$ lies in the *variety* defined by r, denoted $\mathbf{V}(r)$, which is the set of all points $\overline{\alpha} \in \mathbb{F}^m$ such that $r(\overline{\alpha}) = 0$.

On the other hand, the second condition $r(0, \ldots, 0) = 1$ implies that $(0, \ldots, 0)$ is not in the variety $\mathbf{V}(r)$ defined by r. Thus, the hypersurface $\mathbf{V}(r)$ serves as a geometric certificate that the point $(0, \ldots, 0)$ lies outside the image $\mathrm{im}(\mathbf{f})$ of the map $\mathbf{f}$. One can go further, noting that $\mathbf{V}(r)$ is a certificate that $(0, \ldots, 0)$ lies outside the closure of $\mathrm{im}(\mathbf{f})$ in the Zariski topology, but we will not dwell on this.

For a polynomial map $\mathbf{f}$, one can view a Geometric IPS refutation r as an annihilator of $\mathbf{f}$ that is additionally required to have a nonzero constant term. The definition of Geometric IPS specifies that the constant term is 1, but the precise constant is not important. We further explore the connection between annihilators and Geometric IPS refutations in Section 5.

Our focus will be on restricted subsystems of IPS and Geometric IPS. For a complexity class $\mathscr{C}$, we use $\mathscr{C}$-IPS to refer to IPS with the additional restriction that the refutations are computable in $\mathscr{C}$, and likewise for Geometric $\mathscr{C}$-IPS. The formal definition appears below.

Definition 2.14 ($\mathscr{C}$-IPS, Geometric $\mathscr{C}$-IPS)**.** Let $\mathcal{F} = (\mathcal{F}_n)_{n \in \mathbb{N}}$ be a family of systems of polynomial equations. Let $\mathscr{C}$ be a complexity class. We say that $\mathcal{F}$ *can be refuted by $\mathscr{C}$-IPS* if there is a p-family $(r_n)_{n \in \mathbb{N}} \in \mathscr{C}$ such that r_n is an IPS refutation of $\mathcal{F}_n$. If r_n is a Geometric IPS refutation of $\mathcal{F}_n$, we say that $\mathcal{F}$ *can be refuted by Geometric $\mathscr{C}$-IPS*. ◊

As mentioned in the introduction, we will be particularly interested in proving lower bounds for $\mathscr{C}$-IPS where the hard system of polynomial equations can be computed in some weaker class $\mathscr{D} \subsetneq \mathscr{C}$. We formalize this with the notion of $\mathscr{D}$-equations below.

Definition 2.15 ($\mathscr{D}$-equations)**.** Let $\mathcal{F} = (\mathcal{F}_n)_{n \in \mathbb{N}}$ be a family of systems of polynomial equations, where

$$\mathcal{F}_n = \{f_{n,1}(\overline{x}) = 0, \ldots, f_{n,m(n)}(\overline{x}) = 0\}.$$

Let $\mathscr{D}$ be a complexity class. We say that $\mathcal{F}$ is a *family of $\mathscr{D}$-equations* if every p-family of the form $(f_{n,i_n})_{n \in \mathbb{N}}$ is in $\mathscr{D}$. ◊

2.4 The Jacobian Criterion

We will need to compute the transcendence degree of sets of polynomials, which we can do over fields of characteristic zero using the Jacobian Criterion. To state the Jacobian Criterion, we first need to define the Jacobian of a set of polynomials.

Definition 2.16 (Jacobian)**.** Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$. The *Jacobian* of $f_1, \ldots, f_m$, denoted $\operatorname{Jac}(f_1, \ldots, f_m)$, is the $m \times n$ matrix whose (i, j) entry is given by

$$\operatorname{Jac}(f_1, \ldots, f_m)_{i,j} \coloneqq \frac{\partial f_i}{\partial x_j}. \qquad \Diamond$$

We now state the Jacobian Criterion, which gives a precise characterization of transcendence degree over fields of characteristic zero.

Theorem 2.17 (Jacobian Criterion [Jac41], see [ER93])**.** *Let $\mathbb{F}$ be a field of characteristic zero. Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$. Then the transcendence degree of $\{f_1, \ldots, f_m\}$ satisfies*

$$\operatorname{trdeg}_{\mathbb{F}}(f_1, \ldots, f_m) = \operatorname{rank}_{\mathbb{F}(\overline{x})} \operatorname{Jac}(f_1, \ldots, f_m).$$

As the following proposition shows, the rank of the Jacobian still provides a lower bound on the transcendence degree of a set of polynomials when working over a field of positive characteristic.

Proposition 2.18 (see [DGW09, Section 3])**.** *Let $\mathbb{F}$ be an arbitrary field. Let $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$. Then the transcendence degree of $\{f_1, \ldots, f_m\}$ satisfies*

$$\operatorname{trdeg}_{\mathbb{F}}(f_1, \ldots, f_m) \geqslant \operatorname{rank}_{\mathbb{F}(\overline{x})} \operatorname{Jac}(f_1, \ldots, f_m).$$

2.5 The Resultant

We now define the resultant, a useful tool in polynomial factorization and elimination theory.

Definition 2.19 (Resultant)**.** Let $\mathbb{F}$ be a field. Let $f(x) = \sum_{i=0}^{n} f_i x^i$ and $g(x) = \sum_{i=0}^{m} g_i x^i$ be univariate polynomials in $\mathbb{F}[x]$ of degrees n and m, respectively. The *resultant* of f and g, denoted by $\operatorname{res}(f, g)$, is given by

$$\operatorname{res}(f, g) \coloneqq \det \begin{pmatrix} f_n & & & & g_m & & & \\ f_{n-1} & f_n & & & g_{m-1} & g_m & & \\ \vdots & \vdots & \ddots & & \vdots & \vdots & \ddots & \\ \vdots & \vdots & & f_n & g_1 & \vdots & & g_m \\ f_0 & \vdots & & f_{n-1} & g_0 & \vdots & & \vdots \\ & f_0 & & \vdots & & g_0 & & \vdots \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & f_0 & & & & g_0 \end{pmatrix},$$

where the matrix above is an $(n + m) \times (n + m)$ matrix whose first m columns are formed from the coefficients of f and whose last n columns are formed from the coefficients of g. $\Diamond$

As the following lemma shows, the resultant can be used to check if two polynomials share a common factor.

Lemma 2.20 (see, e.g., [vzGG13, Corollary 6.17]). *Let $\mathbb{F}$ be a field and let $f, g \in \mathbb{F}[x]$. Then $\operatorname{res}(f, g) = 0$ if and only if f and g have a nontrivial common factor in $\mathbb{F}[x]$.*

We will also make use of resultants of multivariate polynomials. If $f, g \in \mathbb{F}[x_1, \ldots, x_n, y]$ are multivariate polynomials, we may regard them instead as univariate polynomials in y with coefficients that are themselves polynomials in $x_1, \ldots, x_n$. We write $\operatorname{res}_y(f(\overline{x}, y), g(\overline{x}, y))$ for the corresponding resultant, which is a polynomial in $x_1, \ldots, x_n$. The resultant $\operatorname{res}_y(f, g)$ can be used to test if f and g share a common factor. However, this resultant only detects common factors that depend on the variable y.

Lemma 2.21 (see, e.g., [vzGG13, Corollary 6.20]). *Let $\mathbb{F}$ be a field and let $f, g \in \mathbb{F}[x_1, \ldots, x_n, y]$. Then $\operatorname{res}_y(f, g) = 0$ if and only if f and g have a nontrivial common factor $h \in \mathbb{F}[x_1, \ldots, x_n, y]$ that depends on the variable y.*

Although it is immediate from the definition that $\operatorname{res}_y(f, g)$ does not depend on the variable y, the following proposition says that the resultant eliminates y from f and g in a structured manner.

Proposition 2.22 (see, e.g., [vzGG13, Corollary 6.21]). *Let $\mathbb{F}$ be a field and let $f, g \in \mathbb{F}[x_1, \ldots, x_n, y]$. Then the resultant $\operatorname{res}_y(f(\overline{x}, y), g(\overline{x}, y))$ is an element of the elimination ideal $\langle f, g \rangle \cap \mathbb{F}[x_1, \ldots, x_n]$.*

3 A VNC^0-Computable Generator

In this section, we describe and analyze a construction of a VNC^0-computable hitting set generator $\mathcal{G}$. The generator $\mathcal{G}$ is defined using an arithmetic circuit Φ that computes a polynomial $f(\overline{x})$. We will obtain a complete description of the annihilator ideal $\operatorname{Ann}(\mathcal{G})$: the ideal $\operatorname{Ann}(\mathcal{G})$ is principal and is generated by a polynomial that is closely related to the polynomial $f(\overline{x})$ used to define the map $\mathcal{G}$. This structure allows us to infer pseudorandom properties of $\mathcal{G}$ from hardness of multiples of $f(\overline{x})$.

3.1 Local Encodings of Circuits

We begin by describing the construction of our generator.

Definition 3.1 (Local encoding). Let Φ be an n-variate arithmetic circuit of fan-in two and size s. Let $\overline{\alpha} \in \mathbb{F}^n$ and let $\beta \in \mathbb{F}$. The *local encoding of* $\Phi(\overline{\alpha}) = \beta$ is the polynomial map $\mathcal{G} : \mathbb{F}^{n+s} \to \mathbb{F}^{n+s+1}$ defined by

$$\mathcal{G}(x_1, \ldots, x_n, y_1, \ldots, y_s) \coloneqq (\mathcal{G}_{\text{input}}(\overline{x}, \overline{y}), \mathcal{G}_{\text{internal}}(\overline{x}, \overline{y}), \mathcal{G}_{\text{output}}(\overline{x}, \overline{y})),$$

where each of the polynomial maps $\mathcal{G}_{\text{input}}$, $\mathcal{G}_{\text{internal}}$, and $\mathcal{G}_{\text{output}}$ are defined below.

1. The polynomial map $\mathcal{G}_{\text{input}} : \mathbb{F}^{n+s} \to \mathbb{F}^n$ is given by

$$\mathcal{G}_{\text{input}}(\overline{x}, \overline{y}) \coloneqq (x_1 - \alpha_1, \ldots, x_n - \alpha_n).$$

2. The map $\mathcal{G}_{\text{internal}} : \mathbb{F}^{n+s} \to \mathbb{F}^s$ is defined as follows. Let V be the set of gates of Φ. Let $v_1, \ldots, v_s$ be a topological ordering of the internal gates of Φ, where v_s is the output gate of Φ. Define the function $L : V \to \mathbb{F} \cup \{x_1, \ldots, x_n, y_1, \ldots, y_s\}$ via

$$L(v) \coloneqq \begin{cases} \gamma & \text{if } v \text{ is an input gate labeled by a constant } \gamma \in \mathbb{F}, \\ x_i & \text{if } v \text{ is an input gate labeled by the variable } x_i, \\ y_i & \text{if } v \text{ is the } i^{\text{th}} \text{ internal gate in topological order.} \end{cases}$$

   The i^{th} output of $\mathcal{G}_{\text{internal}}$ is defined in terms of the i^{th} internal gate v_i and its children u and w.

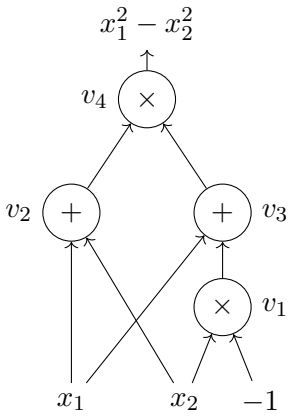

Figure 1: An arithmetic circuit computing the polynomial $x_1^2 - x_2^2$.

- If v_i is an addition gate, then the i^{th} output of $\mathcal{G}_{\text{internal}}$ is the polynomial $L(v_i) - (L(u) + L(w))$.
- If v_i is a product gate, then the i^{th} output of $\mathcal{G}_{\text{internal}}$ is the polynomial $L(v_i) - L(u)L(w)$.

3. The polynomial map $\mathcal{G}_{\text{output}} : \mathbb{F}^{n+s} \to \mathbb{F}$ is given by

$$\mathcal{G}_{\text{output}}(\overline{x}, \overline{y}) \coloneqq y_s - \beta. \qquad \lozenge$$

The name "local encoding" arises from the fact that the local encoding $\mathcal{G}$ of $\Phi(\overline{\alpha}) = \beta$ corresponds to a system of polynomial equations that is satisfiable if and only if the circuit Φ outputs β when evaluated at $\overline{\alpha}$. The corresponding system of polynomial equations is $\mathcal{G}(\overline{x}, \overline{y}) = (0, \ldots, 0)$. The first n equations $\mathcal{G}_{\text{input}}(\overline{x}, \overline{y}) = (0, \ldots, 0)$ ensure that the input to the circuit Φ is the point $\overline{\alpha}$. The next s equations $\mathcal{G}_{\text{internal}}(\overline{x}, \overline{y}) = (0, \ldots, 0)$ enforce that for each $i \in [s]$, the value of the variable y_i equals the value of the i^{th} internal gate when Φ is evaluated at $\overline{\alpha}$. Finally, the equation $\mathcal{G}_{\text{output}}(\overline{x}, \overline{y}) = 0$ expresses that the output gate of Φ evaluates to β.

The following lemma describes the parameters of a local encoding, which follow immediately from Definition 3.1.

Lemma 3.2. *Let Φ be an n-variate arithmetic circuit of size s and fan-in two, and let $\overline{\alpha} \in \mathbb{F}^n$ and $\beta \in \mathbb{F}$. Let $\mathcal{G} : \mathbb{F}^{n+s} \to \mathbb{F}^{n+s+1}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$. Then the following hold.*

1. *The seed length of $\mathcal{G}$ is $n + s$ and the stretch of $\mathcal{G}$ is 1.*
2. *The degree of $\mathcal{G}$ is 2.*
3. *Every output of $\mathcal{G}$ can be computed by an arithmetic formula of size 2.*

In the subsections to come, we determine the annihilator ideals of local encodings and use this to infer pseudorandom properties of local encodings when the circuit Φ computes a sufficiently-hard polynomial. Before moving on to these general results, we first work out a small, concrete example to build intuition for how annihilators of local encodings behave.

Example 3.3. Let Φ be the arithmetic circuit depicted in Figure 1 that computes the polynomial $x_1^2 - x_2^2$. The internal gates of Φ are labeled as v_1, v_2, v_3, and v_4, corresponding to a topological

ordering of the internal gates. For $\alpha_1, \alpha_2, \beta \in \mathbb{F}$, the local encoding of $\Phi(\alpha_1, \alpha_2) = \beta$ is the polynomial map $\mathcal{G} : \mathbb{F}^6 \to \mathbb{F}^7$ given by

$$\mathcal{G}(x_1, x_2, y_1, y_2, y_3, y_4) = (x_1 - \alpha_1, x_2 - \alpha_2, y_1 + x_2, y_2 - x_1 - x_2, y_3 - x_1 - y_1, y_4 - y_2 y_3, y_4 - \beta).$$

What do the annihilators of $\mathcal{G}$ look like? One annihilator can be constructed by writing y_4 as a polynomial combination of the first 6 outputs of $\mathcal{G}$ and then taking the difference of this polynomial and $z_7 + \beta$. To do this, we iteratively find polynomials $h_i \in \mathbb{F}[z_1, \ldots, z_7]$ so that $(h_i \circ \mathcal{G})(\overline{x}, \overline{y}) = y_i$. The desired polynomials h_1, h_2, h_3, and h_4 are given by

$$\begin{aligned} h_1(\overline{z}) &= z_3 - (z_2 + \alpha_2) \\ h_2(\overline{z}) &= z_4 + (z_1 + \alpha_1) + (z_2 + \alpha_2) \\ h_3(\overline{z}) &= z_5 + (z_1 + \alpha_1) + h_1(\overline{z}) \\ h_4(\overline{z}) &= z_6 + h_2(\overline{z}) h_3(\overline{z}). \end{aligned}$$

Because $(h_4 \circ \mathcal{G})(\overline{x}, \overline{y}) = y_4$, it follows that

$$h(\overline{z}) \coloneqq h_4(\overline{z}) - (z_7 + \beta)$$

is a nonzero annihilator of $\mathcal{G}$. Because $\mathrm{Ann}(\mathcal{G})$ is an ideal, every multiple of h is also an annihilator of $\mathcal{G}$. Using computer software, such as Macaulay2, one can verify that these are the only annihilators of $\mathcal{G}$: the ideal $\mathrm{Ann}(\mathcal{G})$ is precisely the principal ideal generated by h.

To prove that local encodings are pseudorandom, we need to understand the complexity of their annihilators. As a first step towards this, let's understand how the complexity of h relates to the complexity of $x_1^2 - x_2^2$, the polynomial computed by Φ. Expanding out $h(\overline{z})$ as

$$\begin{aligned} h(\overline{z}) = z_1^2 - z_2^2 + z_1 z_3 + z_2 z_3 + z_1 z_4 - z_2 z_4 + z_3 z_4 + z_1 z_5 + z_2 z_5 + z_4 z_5 + 2\alpha_1 z_1 - 2\alpha_2 z_2 \\ + (\alpha_1 + \alpha_2) z_3 + (\alpha_1 - \alpha_2) z_4 + (\alpha_1 + \alpha_2) z_5 + z_6 - z_7 + \alpha_1^2 - \alpha_2^2 - \beta, \end{aligned}$$

we rewrite $h(\overline{z})$ as

$$h(\overline{z}) = \big((z_1 + \alpha_1)^2 - (z_2 + \alpha_2)^2\big) + g(\overline{z}) - z_7 - \beta,$$

where

$$\begin{aligned} g(\overline{z}) \coloneqq z_1 z_3 + z_2 z_3 + z_1 z_4 - z_2 z_4 + z_3 z_4 + z_1 z_5 + z_2 z_5 + z_4 z_5 \\ + (\alpha_1 + \alpha_2) z_3 + (\alpha_1 - \alpha_2) z_4 + (\alpha_1 + \alpha_2) z_5 + z_6. \end{aligned}$$

Importantly, the polynomial g is an element of the ideal $\langle z_3, z_4, z_5, z_6 \rangle \subseteq \mathbb{F}[\overline{z}]$. In other words, every monomial of g is divisible by one of the variables z_3, z_4, z_5, or z_6. By setting $z_3 = \cdots = z_7 = 0$, we see that h projects to $\big((z_1 + \alpha)^2 - (z_2 + \alpha_2)^2\big) - \beta$. Applying the change of variables $(z_1, z_2) \mapsto (z_1 - \alpha_1, z_2 - \alpha_2)$, this projection of h becomes $\big(z_1^2 - z_2^2\big) - \beta$. Adding β yields $z_1^2 - z_2^2$, the polynomial computed by the circuit Φ.

By zeroing out some variables, shifting other variables by a constant, and adding an appropriate constant, the generator h is transformed to $z_1^2 - z_2^2$, the polynomial that the circuit Φ computes. This transformation of h has low complexity, so a small circuit that computes h can be used to compute $z_1^2 - z_2^2$ with similar complexity. In the contrapositive, a lower bound on the complexity of $z_1^2 - z_2^2$ implies a comparable lower bound on h. This reduction is not particularly useful for the specific example at hand, as the polynomial $z_1^2 - z_2^2$ is easy to compute.

This example illustrates some features of the general case. The ideal $\mathrm{Ann}(\mathcal{G})$ is always principal (Lemma 3.4) and is generated by a polynomial with the same structure as h above (Proposition 3.6). Using this structure, we can prove lower bounds on the complexity of h and its multiples by appealing to lower bounds on the complexity of the polynomial computed by Φ and its multiples (Lemma 3.7). $\lozenge$

3.2 Annihilators of Local Encodings

Having defined our candidate generator $\mathcal{G}$ as the local encoding of $\Phi(\overline{\alpha}) = \beta$ for an arithmetic circuit Φ, we now proceed to analyze its annihilator ideal $\mathrm{Ann}(\mathcal{G})$. Our ultimate goal is to prove lower bounds on the complexity of every nonzero polynomial in $\mathrm{Ann}(\mathcal{G})$, as this equates (via Lemma 2.11) to proving that $\mathcal{G}$ hits some circuit class. To do this, we first need to understand what the polynomials in $\mathrm{Ann}(\mathcal{G})$ look like.

We begin by showing that $\mathrm{Ann}(\mathcal{G})$ is a nonzero principal ideal. This means that there is a single polynomial $h(\overline{z})$ such that every element of $\mathrm{Ann}(\mathcal{G})$ is a multiple of h. This makes the task of proving lower bounds for annihilators much easier, as we only have to reason about multiples of a single polynomial.

The following lemma is essentially due to Kayal [Kay09, Lemma 7]. We cannot directly apply [Kay09, Lemma 7], as the polynomials in a local encoding do not necessarily fit the hypothesis of the lemma as it appears in Kayal's work. However, inspecting the proof, it is clear that Kayal's argument applies to our setting. We include a (nearly identical) proof for the sake of completeness.

Lemma 3.4. *Let Φ be an n-variate arithmetic circuit of size s and let $\mathcal{G}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$. Then the ideal $\mathrm{Ann}(\mathcal{G})$ is a nonzero principal ideal.*

Proof. The polynomial map $\mathcal{G}$ consists of $n + s + 1$ polynomials in $n + s$ variables. Because there are more polynomials than there are variables, the outputs of $\mathcal{G}$ are algebraically dependent, so the annihilator ideal $\mathrm{Ann}(\mathcal{G})$ is nonempty.

Let $a(z_1, \ldots, z_{n+s+1}) \in \mathrm{Ann}(\mathcal{G})$ be a nonzero element of minimal degree. We claim that $a(\overline{z})$ generates the ideal $\mathrm{Ann}(\mathcal{G})$. To prove this, it suffices to show that for any nonzero $b(\overline{z}) \in \mathrm{Ann}(\mathcal{G})$, the polynomial $a(\overline{z})$ divides $b(\overline{z})$.

We first show that $a(\overline{z})$ is irreducible. Suppose $a(\overline{z})$ factors as $a(\overline{z}) = a_1(\overline{z})a_2(\overline{z})$. Then we have

$$0 = a(\mathcal{G}(\overline{x}, \overline{y})) = a_1(\mathcal{G}(\overline{x}, \overline{y})) \cdot a_2(\mathcal{G}(\overline{x}, \overline{y})),$$

so either $a_1(\mathcal{G}(\overline{x}, \overline{y})) = 0$ or $a_2(\mathcal{G}(\overline{x}, \overline{y})) = 0$. Suppose, without loss of generality, that $a_1(\mathcal{G}(\overline{x}, \overline{y})) = 0$. Then by definition we have $a_1(\overline{z}) \in \mathrm{Ann}(\mathcal{G})$. Because a was chosen to be a nonzero element of $\mathrm{Ann}(\mathcal{G})$ of minimal degree, we have $\deg(a) \leqslant \deg(a_1)$. On the other hand, because a_1 divides a, we have $\deg(a_1) \leqslant \deg(a)$. Together, these inequalities imply $\deg(a) = \deg(a_1)$. It follows that $\deg(a_2) = 0$, so $a_2(\overline{z})$ is a nonzero constant polynomial. Hence $a(\overline{z})$ is irreducible as claimed.

Now let $b(\overline{z}) \in \mathrm{Ann}(\mathcal{G})$ be nonzero. To see that $a(\overline{z})$ divides $b(\overline{z})$, consider their resultant with respect to z_{n+s+1},

$$r(\overline{z}) \coloneqq \mathrm{res}_{z_{n+s+1}}(a(\overline{z}), b(\overline{z})).$$

By Proposition 2.22, we have

$$r(\overline{z}) \in \langle a(\overline{z}), b(\overline{z}) \rangle \cap \mathbb{F}[z_1, \ldots, z_{n+s}] \subseteq \mathrm{Ann}(\mathcal{G}) \cap \mathbb{F}[z_1, \ldots, z_{n+s}].$$

This implies $r(\overline{z})$ vanishes on the first $n + s$ outputs of $\mathcal{G}$. If $r(\overline{z})$ were nonzero, then this vanishing would imply that the first $n+s$ outputs of $\mathcal{G}$ are algebraically dependent. However, this is impossible: the Jacobian of the first $n + s$ outputs of $\mathcal{G}$ is a triangular matrix with ones along the diagonal, so Proposition 2.18 implies they are algebraically independent. Thus, it follows that $r(\overline{z}) = 0$. Lemma 2.21 implies that $a(\overline{z})$ and $b(\overline{z})$ have a common factor. Because $a(\overline{z})$ is irreducible, it follows that $a(\overline{z})$ divides $b(\overline{z})$ as desired. □

Our next goal is to describe a polynomial that generates the ideal $\mathrm{Ann}(\mathcal{G})$. Just as in Example 3.3, we can find a dependency between the outputs of $\mathcal{G}$ by iteratively expressing each of the y variables

as a polynomial combination of the first $n+s$ outputs of $\mathcal{G}$. Doing this results in a polynomial $h_s(\overline{z})$ that satisfies $h_s(\mathcal{G}(\overline{x},\overline{y})) = y_s$. Subtracting $z_{n+s+1} + \beta$ from $h_s(\overline{z})$ results in an annihilator of $\mathcal{G}$. It turns out that the annihilator obtained in this way is irreducible, and hence it generates the ideal $\mathrm{Ann}(\mathcal{G})$.

We start with the following lemma, which shows that each of the variables $y_1, \ldots, y_s$ can be obtained as a polynomial combination of the first $n+s$ outputs of $\mathcal{G}$. These polynomials are obtained by simulating the underlying circuit Φ in a gate-by-gate manner. As a byproduct of this construction, we also obtain a bound on the circuit complexity of these polynomials.

Lemma 3.5. *Let Φ be an n-variate arithmetic circuit of size s and let $\mathcal{G}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$. For every $k \in [s]$, there is a multi-output arithmetic circuit Ψ_k such that the following hold.*

- *The size of Ψ_k is bounded by $O(k)$.*
- *The circuit Ψ_k outputs k polynomials $h_1, \ldots, h_k \in \mathbb{F}[z_1, \ldots, z_{n+s+1}]$ that depend only on the variables $z_1, \ldots, z_{n+k}$. For each $i \in [k]$, the polynomial h_i satisfies the identity $h_i(\mathcal{G}(\overline{x},\overline{y})) = y_i$.*

Proof. We proceed by induction on k. Let Ψ_{k-1} be the circuit given by induction. (If $k = 1$, then Ψ_0 is the empty circuit.) To construct Ψ_k, it suffices to add a constant number of new gates to Ψ_{k-1} in order to compute the desired polynomial $h_k(\overline{z})$.

To avoid a multitude of sub-cases, we need some additional notation. Let V be the set of gates of Φ. Recall the function $L : V \to \mathbb{F} \cup \{x_1, \ldots, x_n, y_1, \ldots, y_s\}$ used in the definition of $\mathcal{G}$, which was given by

$$L(v) \coloneqq \begin{cases} \gamma & \text{if } v \text{ is an input gate labeled by a constant } \gamma \in \mathbb{F}, \\ x_i & \text{if } v \text{ is an input gate labeled by the variable } x_i, \\ y_i & \text{if } v \text{ is the } i^{\text{th}} \text{ internal gate in topological order.} \end{cases}$$

Let $V_{\leqslant k-1}$ be the set consisting of the input gates and first $k-1$ internal gates of Φ. Define the function $\hat{L} : V_{\leqslant k-1} \to \mathbb{F}[\overline{z}]$ via

$$\hat{L}(v) \coloneqq \begin{cases} \gamma & \text{if } v \text{ is an input gate labeled by a constant } \gamma \in \mathbb{F}, \\ z_i + \alpha_i & \text{if } v \text{ is an input gate labeled by the variable } x_i, \\ h_i(\overline{z}) & \text{if } v \text{ is the } i^{\text{th}} \text{ internal gate in topological order.} \end{cases}$$

Note that every output of $\hat{L}$ is either already computed by the circuit Ψ_{k-1} or can be computed using a constant number of gates. An immediate consequence of the inductive hypothesis is that the polynomial $\hat{L}(v) \circ \mathcal{G}(\overline{x},\overline{y})$ is given by

$$\hat{L}(v) \circ \mathcal{G}(\overline{x},\overline{y}) = \begin{cases} \gamma & \text{if } v \text{ is an input gate labeled by a constant } \gamma \in \mathbb{F}, \\ x_i & \text{if } v \text{ is an input gate labeled by the variable } x_i, \\ y_i & \text{if } v \text{ is the } i^{\text{th}} \text{ internal gate in topological order.} \end{cases}$$

Put more succinctly, we have the identity

$$\hat{L}(v) \circ \mathcal{G}(\overline{x},\overline{y}) = L(v)$$

for all gates $v \in V_{\leqslant k-1}$.

We now describe how to compute the desired polynomial $h_k(\overline{z})$. Let v be the k^{th} internal gate of Φ, and let u and w be its children. There are two cases to consider, depending on whether v is an addition or multiplication gate.

- Suppose $v = u + w$. Define
$$h_k(\overline{z}) \coloneqq z_{n+k} + \hat{L}(u) + \hat{L}(w).$$
It is clear that $h_k(\overline{z})$ can be computed by adding $O(1)$ gates to Ψ_{k-1}, so the size of Ψ_k is bounded by $O(k)$ as claimed.
It remains to verify that h_k satisfies the claimed identity. Composing with $\mathcal{G}(\overline{x}, \overline{y})$, we have
$$\begin{aligned} h_k(\mathcal{G}(\overline{x}, \overline{y})) &= y_k - L(u) - L(w) + \hat{L}(u) \circ \mathcal{G}(\overline{x}, \overline{y}) + \hat{L}(w) \circ \mathcal{G}(\overline{x}, \overline{y}) \\ &= y_k - L(u) - L(w) + L(u) + L(w) \\ &= y_k \end{aligned}$$
as desired.

- Suppose $v = u \times w$. Define
$$h_k(\overline{z}) \coloneqq z_{n+k} + \hat{L}(u)\hat{L}(w).$$
As in the previous case, it is clear that $h_k(\overline{z})$ can be computed by adding $O(1)$ gates to Ψ_{k-1}, so the size of Ψ_k is bounded by $O(k)$.
We now verify the claimed identity on h_k. We compose with $\mathcal{G}(\overline{x}, \overline{y})$ to obtain
$$\begin{aligned} h_k(\mathcal{G}(\overline{x}, \overline{y})) &= y_k - L(u)L(w) + (\hat{L}(u) \circ \mathcal{G}(\overline{x}, \overline{y})) \cdot (\hat{L}(w) \circ \mathcal{G}(\overline{x}, \overline{y})) \\ &= y_k - L(u)L(w) + L(u)L(w) \\ &= y_k \end{aligned}$$
as desired.

In both cases, we can construct Ψ_k by adding $O(1)$ gates to Ψ_{k-1}. □

Using the preceding lemma, we now describe the generator of the ideal $\mathrm{Ann}(\mathcal{G})$.

Proposition 3.6. *Let Φ be an n-variate arithmetic circuit of size s that computes a polynomial $f(x_1, \ldots, x_n)$. Let $\mathcal{G}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$. Then there is a polynomial $h \in \mathbb{F}[z_1, \ldots, z_{n+s+1}]$ such that the following hold.*

1. *The ideal $\mathrm{Ann}(\mathcal{G})$ is generated by $h(\overline{z})$.*
2. *There exists a polynomial $g \in \mathbb{F}[z_1, \ldots, z_{n+s}]$, not depending on the variable z_{n+s+1}, such that $g \in \langle z_{n+1}, \ldots, z_{n+s} \rangle$ and*
$$h(\overline{z}) = z_{n+s+1} - f(z_1 + \alpha_1, \ldots, z_n + \alpha_n) + g(\overline{z}) + \beta.$$
3. *There is an arithmetic circuit of size $O(s)$ that computes $h(\overline{z})$.*

Proof. Let Ψ_s be the circuit of size $O(s)$ obtained by applying Lemma 3.5 to Φ and $\mathcal{G}$. Let $h_s(\overline{z})$ be the output of Ψ_s that satisfies $h_s(\mathcal{G}(\overline{x}, \overline{y})) = y_s$. Define
$$h(\overline{z}) \coloneqq z_{n+s+1} - h_s(\overline{z}) + \beta.$$
It is clear from this definition that $h(\overline{z})$ can be computed by an arithmetic circuit of size $O(s)$. It remains to verify that h generates $\mathrm{Ann}(\mathcal{G})$ and that h has the claimed form.

We first establish that h generates the ideal $\mathrm{Ann}(\mathcal{G})$. To see that $h \in \mathrm{Ann}(\mathcal{G})$, we compute

$$\begin{aligned} h(\mathcal{G}(\overline{x},\overline{y})) &= y_s - \beta - h_s(\mathcal{G}(\overline{x},\overline{y})) + \beta \\ &= y_s - \beta - y_s + \beta \\ &= 0. \end{aligned}$$

Thus $h \in \mathrm{Ann}(\mathcal{G})$. To see that h generates $\mathrm{Ann}(\mathcal{G})$, note that because $h_s(\overline{z})$ does not depend on the variable z_{n+s+1}, the polynomial h is monic and has degree 1 in z_{n+s+1}. This implies that h is irreducible. Because h is irreducible, the ideal $\mathrm{Ann}(\mathcal{G})$ is principal, and $h \in \mathrm{Ann}(\mathcal{G})$, it follows that h generates $\mathrm{Ann}(\mathcal{G})$.

It remains to show that h has the claimed form. Recall that $\mathcal{G}$ is the concatenation of the three polynomial maps $\mathcal{G}_{\mathrm{input}}$, $\mathcal{G}_{\mathrm{internal}}$, and $\mathcal{G}_{\mathrm{output}}$. Because $h \in \mathrm{Ann}(\mathcal{G})$, we have the identity

$$h(\mathcal{G}_{\mathrm{input}}(\overline{x},\overline{y}), \mathcal{G}_{\mathrm{internal}}(\overline{x},\overline{y}), \mathcal{G}_{\mathrm{output}}(\overline{x},\overline{y})) = 0.$$

Let $f_i(\overline{x})$ be the polynomial computed by the i^{th} internal gate of Φ and consider the substitution given by $y_i \mapsto f_i(\overline{x})$. We claim that under this substitution, the outputs of $\mathcal{G}_{\mathrm{internal}}$ are mapped to zero. Indeed, if the i^{th} gate of Φ is an addition gate whose children are the j^{th} and k^{th} internal gates, then under this substitution the i^{th} output of $\mathcal{G}_{\mathrm{internal}}$ becomes

$$f_i(\overline{x}) - f_j(\overline{x}) - f_k(\overline{x}) = f_i(\overline{x}) - f_i(\overline{x}) = 0.$$

Likewise, if the i^{th} internal gate of Φ is a multiplication gate whose children are the j^{th} and k^{th} internal gates, then the i^{th} output of $\mathcal{G}_{\mathrm{internal}}$ is mapped to

$$f_i(\overline{x}) - f_j(\overline{x}) f_k(\overline{x}) = f_i(\overline{x}) - f_i(\overline{x}) = 0.$$

Similar cancellations occur when the i^{th} internal gate has one or more input gates as children; we omit the straightforward calculations.

Applying the substitution $y_i \mapsto f_i(\overline{x})$ to the equation $h(\mathcal{G}(\overline{x},\overline{y})) = 0$, we obtain the identity

$$h(x_1 - \alpha_1, \ldots, x_n - \alpha_n, 0, \ldots, 0, f(\overline{x}) - \beta) = 0.$$

Next, we apply the shift $x_i \mapsto x_i + \alpha_i$ to obtain

$$h(x_1, \ldots, x_n, 0, \ldots, 0, f(\overline{x} + \overline{\alpha}) - \beta) = 0.$$

This implies that $z_{n+s+1} - f(\overline{x} + \overline{\alpha}) + \beta$ is a factor of $h(x_1, \ldots, x_n, 0, \ldots, 0, z_{n+s+1})$. Because the polynomial $h(x_1, \ldots, x_n, 0, \ldots, 0, z_{n+s+1})$ is monic and degree 1 in z_{n+s+1}, we obtain the equality

$$h(x_1, \ldots, x_n, 0, \ldots, 0, z_{n+s+1}) = z_{n+s+1} - f(\overline{x} + \overline{\alpha}) + \beta.$$

This implies

$$h(\overline{z}) = z_{n+s+1} - f(\overline{z} + \overline{\alpha}) + g(\overline{z}) + \beta,$$

where $g \in \mathbb{F}[\overline{z}]$ is a polynomial that vanishes on the substitution $z_{n+1} = \cdots = z_{n+s} = 0$, which is equivalent to g being an element of the ideal $\langle z_{n+1}, \ldots, z_{n+s} \rangle$. Thus $h(\overline{z})$ has the claimed form. $\square$

3.3 Pseudorandomness from Hardness of Multiples

In this subsection, we investigate the pseudorandom properties of local encodings. Suppose $\mathcal{G}$ is the local encoding of $\Phi(\overline{\alpha}) = \beta$, where Φ is an arithmetic circuit that computes a polynomial $f(\overline{x})$. As we saw in Proposition 3.6, the ideal $\mathrm{Ann}(\mathcal{G})$ is generated by a polynomial $h(\overline{z})$ that is similar to $f(\overline{x})$, in the sense that any circuit computing h can be modified in a simple way to produce a circuit that computes f. This reduction shows that lower bounds on the complexity of f imply lower bounds on the complexity of h.

To show that $\mathcal{G}$ hits a circuit class $\mathscr{C}$, we need to show that every nonzero multiple of h is hard for $\mathscr{C}$, not just that h itself is hard for $\mathscr{C}$. Because the polynomials h and f are closely related, it seems reasonable to expect that lower bounds on the complexity of multiples of f imply comparable lower bounds for multiples of h.

This intuition is correct: suppose $p \in \mathrm{Ann}(\mathcal{G})$ is a polynomial of the form $h(\overline{z}) \cdot r(\overline{z})$. If we work in the ring $\mathbb{F}[z_1, \ldots, z_n][z_{n+1}, \ldots, z_{n+s+1}]$, then the bottom-degree homogeneous component of $h(\overline{z})$ is precisely

$$h_{\mathrm{bot}}(\overline{z}) = -f(z_1 + \alpha_1, \ldots, z_n + \alpha_n) + \beta.$$

Bottom-degree homogeneous components are homomorphic with respect to multiplication, so the bottom-degree component of $h(\overline{z}) \cdot r(\overline{z})$ is a multiple of $h_{\mathrm{bot}}(\overline{z})$. From a circuit computing $h(\overline{z}) \cdot r(\overline{z})$, we can obtain circuit for this multiple of $h_{\mathrm{bot}}(\overline{z})$ by applying polynomial interpolation in the variables $z_{n+1}, \ldots, z_{n+s+1}$. Applying the shift $z_i \mapsto z_i - \alpha_i$ for $i \in [n]$ results in a circuit that computes a multiple of $f(z_1, \ldots, z_n) - \beta$. Thus, if multiples of $f - \beta$ require large circuits, then it must have been the case that the circuit computing $h(\overline{z}) \cdot r(\overline{z})$ was large to begin with, so we obtain a lower bound for all nonzero polynomials in $\mathrm{Ann}(\mathcal{G})$.

We now make the preceding sketch precise, showing that lower bounds on $\mathrm{Ann}(\mathcal{G})$ follow from lower bounds for multiples of $f(\overline{x})$. Although the interpolation argument described above works in general, we state and prove this lemma for three concrete measures of complexity (bounded-depth circuits, formulas, and branching programs) that correspond to the applications we give in Section 4.

Lemma 3.7. *Let $\mathbb{F}$ be an infinite field. Let Φ be an n-variate circuit arithmetic circuit that computes a polynomial $f \in \mathbb{F}[\overline{x}]$. Let $\mathcal{G}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$.*

1. *If every nonzero multiple of $f(\overline{x}) - \beta$ requires depth-Δ circuits of size at least s, then every nonzero polynomial $p \in \mathrm{Ann}(\mathcal{G})$ requires depth-$(\Delta - 2)$ circuits of size at least $\Omega(s^{\frac{1}{\Delta - 1}})$.*
2. *If every nonzero multiple of $f(\overline{x}) - \beta$ requires formulas of size at least s, then every nonzero polynomial $p \in \mathrm{Ann}(\mathcal{G})$ requires formulas of size at least $\Omega(\sqrt{s})$.*
3. *If every nonzero multiple of $f(\overline{x}) - \beta$ requires branching programs of size at least s, then every nonzero polynomial $p \in \mathrm{Ann}(\mathcal{G})$ requires branching programs of size at least $\Omega(\sqrt{s})$.*

Proof. Let $p \in \mathrm{Ann}(\mathcal{G})$ be nonzero. By Proposition 3.6, we know that $p(\overline{z})$ can be written as

$$p(\overline{z}) = (z_{n+s+1} - f(z_1 + \alpha_1, \ldots, z_n + \alpha_n) + g(\overline{z}) + \beta) \cdot r(\overline{z}),$$

where $g \in \langle z_{n+1}, \ldots, z_{n+s} \rangle$ and $r(\overline{z})$ is a nonzero polynomial. Consider the substitution

$$z_i \mapsto \begin{cases} z_i - \alpha_i & \text{if } 1 \leqslant i \leqslant n, \\ w \cdot z_i & \text{if } n+1 \leqslant i \leqslant n+s+1, \end{cases}$$

where w is a fresh variable. Under this substitution, the first factor of p is mapped to

$$-f(z_1, \ldots, z_n) + \beta + O(w),$$

where $O(w)$ indicates a term divisible by w. Let

$$\hat{r}(\overline{z}, w) = \sum_{i=a}^{b} r_i(\overline{z}) w^i$$

be the image of $r(\overline{z})$ under this substitution, where $a \leqslant b$ and $r_a(\overline{z}) \neq 0$. Then the image of $p(\overline{z})$ under this substitution is

$$\hat{p}(\overline{z}, w) \coloneqq (-f(\overline{z}) + \beta) \cdot r_a(\overline{z}) w^a + O(w^{a+1}),$$

where $O(w^{a+1})$ indicates a term divisible by w^{a+1}.

We now obtain a circuit that computes $(-f(\overline{z}) + \beta) \cdot r_a(\overline{z})$ using polynomial interpolation with respect to w. Note that $\hat{p}(\overline{z}, w)$ has degree at most $d \coloneqq \deg(p)$ with respect to w. Let $\gamma_0, \ldots, \gamma_d \in \mathbb{F}$ be distinct field elements. Then there are constants $\zeta_0, \ldots, \zeta_d \in \mathbb{F}$ such that

$$(-f(\overline{z}) + \beta) \cdot r_a(\overline{z}) = \sum_{i=0}^{d} \zeta_i \cdot \hat{p}(\overline{z}, \gamma_i).$$

This expresses a nonzero multiple of $f(\overline{z}) - \beta$ as a sum of $\deg(p) + 1$ copies of $\hat{p}(\overline{z}, w)$, each of which can be obtained in a simple manner from the circuit computing $p(\overline{z})$. In particular, if nonzero multiples of $f(\overline{z}) - \beta$ have high complexity, then a similar lower bound holds for the annihilator $p \in \mathrm{Ann}(\mathcal{G})$ that we started with. The precise details of this bound depends on the complexity measure of interest.

1. Suppose p was computed by a circuit of depth $\Delta - 2$ and size t. Then the resulting expression for $(-f(\overline{z}) + \beta) \cdot r_a(\overline{z})$ can be implemented by a circuit of depth Δ and size $O(t \deg(p))$. Because p was computed by a circuit of depth $\Delta - 2$ and size t, we have the bound $\deg(p) \leqslant t^{\Delta - 2}$. Thus, we have a circuit of depth Δ and size $O(t^{\Delta - 1})$ that computes $(-f(\overline{z}) + \beta) \cdot r_a(\overline{z})$. If every nonzero multiple of $f(\overline{z}) - \beta$ requires depth-Δ circuits of size s, we conclude $t \geqslant \Omega(s^{\frac{1}{\Delta - 1}})$ as desired.

2. Suppose p was computed by a formula of size t. As in the previous case, the expression for $(-f(\overline{z}) + \beta) \cdot r_a(\overline{z})$ can be implemented by a formula of size $O(t \deg(p))$. Because p was computed by a formula of size t, we have the bound $\deg(p) \leqslant t$. Thus, we obtain a formula of size $O(t^2)$ that computes a nonzero multiple of $f(\overline{z}) - \beta$. If such polynomials require formulas of size s, we conclude the desired lower bound of $t \geqslant \Omega(\sqrt{s})$.

3. Suppose p was computed by a branching program of size t. The argument in this case is identical to the preceding argument for formulas, so we again conclude the lower bound $t \geqslant \Omega(\sqrt{s})$. □

As an immediate corollary of the preceding lemma, we see that $\mathcal{G}$ hits circuit classes that require large size to compute multiples of $f(\overline{x}) - \beta$.

Corollary 3.8. *Let $\mathbb{F}$ be an infinite field. Let Φ be an n-variate arithmetic circuit that computes a polynomial $f \in \mathbb{F}[\overline{x}]$. Let $\mathcal{G}$ be the local encoding of $\Phi(\overline{\alpha}) = \beta$.*

1. *Suppose every nonzero multiple of $f(\overline{x}) - \beta$ requires depth-Δ circuits of size at least s. Then $\mathcal{G}$ hits circuits of depth $\Delta - 2$ and size $\varepsilon s^{\frac{1}{\Delta - 1}}$ for a sufficiently small constant $\varepsilon > 0$.*

2. *Suppose every nonzero multiple of $f(\overline{x}) - \beta$ requires formulas of size at least s. Then $\mathcal{G}$ hits formulas of size $\varepsilon\sqrt{s}$ for a sufficiently small constant $\varepsilon > 0$.*

3. *Suppose every nonzero multiple of $f(\overline{x}) - \beta$ requires branching programs of size at least s. Then $\mathcal{G}$ hits branching programs of size $\varepsilon\sqrt{s}$ for a sufficiently small constant $\varepsilon > 0$.*

4 Instantiating the Generator

Having analyzed local encodings and their annihilators, we now move on to constructing hitting set generators using local encodings. As Corollary 3.8 shows, to construct a hitting set generator for a class $\mathscr{C}$ using a local encoding, it suffices to take an encoding of a circuit that computes a polynomial whose multiples are hard for the target class $\mathscr{C}$. In this section, we give three constructions of VNC^0-computable generators using local encodings. One generator hits VAC^0 unconditionally, while the others hit VF and VBP under strong but reasonable hardness assumptions. (In fact, the generators for VAC^0 and VF are local encodings of the same circuit family.)

Although our work up to this point has proceeded over an arbitrary field $\mathbb{F}$, this section will require $\mathbb{F}$ to have characteristic zero or sufficiently large characteristic. We omit the large characteristic case for simplicity. The restriction on the field characteristic stems from the fact that the hitting property for local encodings is inferred from the hardness of multiples of an explicit polynomial $f(\overline{x})$. Over fields of small positive characteristic, it is an open problem to show that if a polynomial $f(\overline{x})$ is hard to compute, then all multiples of f are similarly hard to compute. In particular, over a field of characteristic $p > 0$, it is not known if a circuit lower bound for $f(\overline{x})$ implies a comparable lower bound for $f(\overline{x})^p$. For some limited results in this direction, see Andrews [And20].

4.1 Low-Depth Circuits

In our first application of Corollary 3.8, we construct a VNC^0-computable generator that hits VAC^0. To do this, we use the fact that multiples of the determinant are hard for low-depth circuits. In particular, we make use of the following lower bound.

Theorem 4.1 ([LST21; AF22])**.** *Let $\mathbb{F}$ be a field of characteristic zero. Any depth-Δ circuit that computes a multiple of the $n \times n$ determinant $\det_n(X)$ must be of size at least $n^{(\log n)^{\exp(-O(\Delta))}}$.*

To construct a VNC^0-computable generator for VAC^0, we first combine Theorem 4.1 with Corollary 3.8 to show that low-depth circuits are hit by local encodings of arithmetic circuits that compute the determinant.

Lemma 4.2. *Let $\mathbb{F}$ be a field of characteristic zero. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes the $n \times n$ determinant $\det_n(X)$. Let $(A_n)_{n\in\mathbb{N}}$ be a sequence of $n \times n$ matrices. Let $\mathcal{G}_n : \mathbb{F}^{n^2+s(n)} \to \mathbb{F}^{n^2+s(n)+1}$ be the local encoding of $\Phi_n(A_n) = 0$. Then $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbb{N}}$ is a VNC^0-computable hitting set generator for VAC^0.*

Proof. The fact that $\mathcal{G}$ is VNC^0-computable follows immediately from Lemma 3.2. To show that $\mathcal{G}$ hits VAC^0, we will apply Corollary 3.8 and Theorem 4.1. Theorem 4.1 implies that any circuit of depth Δ computing a multiple of $\det_n(X)$ must have size at least

$$n^{(\log n)^{\exp(-O(\Delta))}}.$$

It follows from Corollary 3.8 that $\mathcal{G}_n$ hits circuits of depth Δ and size

$$n^{\frac{(\log n)^{\exp(-O(\Delta))}}{\Delta+1}}.$$

This implies that $\mathcal{G}$ hits VAC^0, as we now show.

Let $(f_n)_{n\in\mathbb{N}} \in \mathrm{VAC}^0$. By definition, there is a fixed constant Δ such that f_n can be computed by circuits of depth Δ and size $n^{O(1)}$. Let $m \leqslant n^{O(1)}$ be the number of variables in f_n and let q be the largest integer such that $q^2 + s(q) + 1 \leqslant m$. Because $s(q) \leqslant q^{O(1)}$, we have

$$q \geqslant m^{\Omega(1)} \geqslant n^{\Omega(1)}.$$

The map $\mathcal{G}_q$, padded appropriately as in Remark 2.9, hits m-variate circuits of depth Δ and size

$$q^{\frac{(\log q)^{\exp(-O(\Delta))}}{\Delta+1}} \gg n^{O(1)}.$$

Thus, for sufficiently large n, the map $\mathcal{G}_q$ hits f_n, so $\mathcal{G}$ hits VAC^0 as claimed. □

Lemma 4.2 gives a VNC^0-computable generator with one bit of stretch that hits VAC^0. We can improve the stretch of this generator by applying independent copies of the generator in parallel. This improves the stretch of the generator from 1 to $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$.

Theorem 4.3. *Let $\mathbb{F}$ be a field of characteristic zero and let $\varepsilon > 0$ be a fixed constant. There is an explicit sequence of polynomial maps $\mathcal{G} = (\mathcal{G} : \mathbb{F}^{n-n^{1-\varepsilon}} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ such that $\mathcal{G}$ is a VNC^0-computable hitting set generator for VAC^0.*

Proof. Let $\hat{\mathcal{G}} = (\hat{\mathcal{G}}_n : \mathbb{F}^{n-1} \to \mathbb{F}^n)$ be the generator of Lemma 4.2, padded appropriately as in Remark 2.9. We take $\mathcal{G}_n : \mathbb{F}^{n-n^{1-\varepsilon}} \to \mathbb{F}^n$ to be the generator obtained by concatenating $n^{1-\varepsilon}$ independent copies of $\hat{\mathcal{G}}_{n^\varepsilon}$. That is, we set $\mathcal{G}_n$ to be the generator

$$\mathcal{G}_n(\overline{x}, \overline{y}) \coloneqq (\hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(1)}, \overline{y}^{(1)}), \ldots, \hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(n^{1-\varepsilon})}, \overline{y}^{(n^{1-\varepsilon})})),$$

where $\overline{x} = \overline{x}^{(1)} \sqcup \cdots \sqcup \overline{x}^{(n^{1-\varepsilon})}$ is a partition of the x variables and likewise $\overline{y} = \overline{y}^{(1)} \sqcup \cdots \sqcup \overline{y}^{(n^{1-\varepsilon})}$ is a partition of the y variables. The VNC^0-computability of $\hat{\mathcal{G}}$ implies that $\mathcal{G}$ is VNC^0-computable.

The fact that $\mathcal{G}$ hits VAC^0 follows from a straightforward hybrid argument, which we now describe. Suppose there is a p-family $f = (f_n)_{n\in\mathbb{N}} \in \mathrm{VAC}^0$ such that $\mathcal{G}$ fails to hit f. We will use this to construct a p-family $g = (g_n)_{n\in\mathbb{N}} \in \mathrm{VAC}^0$ such that $\hat{\mathcal{G}}$ fails to hit g, which contradicts Lemma 4.2.

For the sake of notational simplicity, assume without of generality that f_n is a polynomial on n variables $z_1, \ldots, z_n$. Let $\overline{z} = \overline{z}^{(1)} \sqcup \cdots \sqcup \overline{z}^{(n^{1-\varepsilon})}$ be a partition of the z variables. Consider the hybrid polynomials

$$\begin{aligned}
f_{n,0}(\overline{x}, \overline{y}, \overline{z}) &\coloneqq f_n(\overline{z}^{(1)}, \overline{z}^{(2)}, \ldots, \overline{z}^{(n^{1-\varepsilon})}) \\
f_{n,1}(\overline{x}, \overline{y}, \overline{z}) &\coloneqq f_n(\hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(1)}, \overline{y}^{(1)}), \overline{z}^{(2)}, \ldots, \overline{z}^{(n^{1-\varepsilon})}) \\
f_{n,2}(\overline{x}, \overline{y}, \overline{z}) &\coloneqq f_n(\hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(1)}, \overline{y}^{(1)}), \hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(2)}, \overline{y}^{(2)}), \ldots, \overline{z}^{(n^{1-\varepsilon})}) \\
&\vdots \\
f_{n,n^{1-\varepsilon}}(\overline{x}, \overline{y}, \overline{z}) &\coloneqq f_n(\hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(1)}, \overline{y}^{(1)}), \hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(2)}, \overline{y}^{(2)}), \ldots, \hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(n^{1-\varepsilon})}, \overline{y}^{(n^{1-\varepsilon})})).
\end{aligned}$$

The fact that $\mathcal{G}$ fails to hit f implies that $f_{n,0}(\overline{x}, \overline{y}, \overline{z}) \neq 0$ and $f_{n,n^{1-\varepsilon}}(\overline{x}, \overline{y}, \overline{z}) = 0$ when n is sufficiently large. Thus, there is some $i \in [n^{1-\varepsilon}]$ such that $f_{n,i-1}(\overline{x}, \overline{y}, \overline{z}) \neq 0$ but $f_{n,i}(\overline{x}, \overline{y}, \overline{z}) = 0$. Because $\mathbb{F}$ is infinite, there are inputs $\overline{\alpha}^{(1)}, \ldots, \overline{\alpha}^{(i-1)}, \overline{\beta}^{(1)}, \ldots, \overline{\beta}^{(i-1)}$, and $\overline{\gamma}^{(i+1)}, \ldots, \overline{\gamma}^{(n^{1-\varepsilon})}$ such that

$$g_n(\overline{z}^{(i)}) \coloneqq f_{n,i-1}(\hat{\mathcal{G}}(\overline{\alpha}^{(1)}, \overline{\beta}^{(1)}), \ldots, \hat{\mathcal{G}}(\overline{\alpha}^{(i-1)}, \overline{\beta}^{(i-1)}), \overline{z}^{(i)}, \overline{\gamma}^{(i+1)}, \ldots, \overline{\gamma}^{(n^{1-\varepsilon})}) \neq 0.$$

Note that g is a projection of f, so $g = (g_n)_{n\in\mathbb{N}}$ is a p-family in VAC^0. Furthermore, we have

$$g_n(\hat{\mathcal{G}}_{n^\varepsilon}(\overline{x}^{(i)}, \overline{y}^{(i)})) = f_{i-1}(\hat{\mathcal{G}}(\overline{\alpha}^{(1)}, \overline{\beta}^{(1)}), \ldots, \hat{\mathcal{G}}(\overline{\alpha}^{(i-1)}, \overline{\beta}^{(i-1)}), \hat{\mathcal{G}}(\overline{x}^{(i)}, \overline{y}^{(i)}), \overline{\gamma}^{(i+1)}, \ldots, \overline{\gamma}^{(n^{1-\varepsilon})}) = 0,$$

so $\hat{\mathcal{G}}$ fails to hit g, contradicting Lemma 4.2. □

4.2 Formulas

Next, we construct a VNC^0-computable generator that hits VF. To do this, we need a polynomial whose multiples all require large formulas. Formulas are not known to be closed under factorization, so it is not clear if formula lower bounds for an explicit family of polynomials imply lower bounds for multiples of the same family. For the special case of the determinant, however, lower bounds on border formula complexity imply comparable lower bounds on the (border) formula complexity of multiples, as the following theorem shows.

Theorem 4.4 ([AF22])**.** *Let $\mathbb{F}$ be a field of characteristic zero. Suppose that any border formula which computes the $n \times n$ determinant $\det_n(X)$ must have size at least $n^{\omega(1)}$. Then any border formula that computes a multiple of $\det_n(X)$ must have size $n^{\omega(1)}$.*

Assuming lower bounds on the border formula complexity of the determinant, we can construct a VNC^0-computable generator with one bit of stretch that hits VF. The proof is essentially the same as in Lemma 4.2; the only difference is that the unconditional lower bound of Theorem 4.1 is replaced by the conditional lower bound from Theorem 4.4.

Lemma 4.5. *Let $\mathbb{F}$ be a field of characteristic zero. Assume that any border formula which computes the $n \times n$ determinant $\det_n(X)$ has size $n^{\omega(1)}$. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes the $n \times n$ determinant $\det_n(X)$. Let $(A_n)_{n\in\mathbb{N}}$ be a sequence of $n \times n$ matrices. Let $\mathcal{G}_n : \mathbb{F}^{n^2+s(n)} \to \mathbb{F}^{n^2+s(n)+1}$ be the local encoding of $\Phi_n(A_n) = 0$. Then $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbb{N}}$ is a VNC^0-computable hitting set generator for* VF*.*

Proof. The fact that $\mathcal{G}_n$ is VNC^0-computable is an immediate consequence of Lemma 3.2. To show that $\mathcal{G}$ hits VF, we will invoke Corollary 3.8, Theorem 4.4, and the assumed lower bound on the border formula complexity of the determinant. Because any border formula that computes the $n \times n$ determinant must have size $n^{\omega(1)}$, Theorem 4.4 implies that any formula computing a multiple of $\det_n(X)$ must have size $t \geqslant n^{\omega(1)}$. It follows from Corollary 3.8 that the map $\mathcal{G}_n$ hits formulas of size $\varepsilon\sqrt{t} \geqslant n^{\omega(1)}$ for some constant $\varepsilon > 0$. In particular, for a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VF}$, the map $\mathcal{G}_{q(n)}$ (where $q(n) \geqslant n^{\Omega(1)}$ is the index of the n-output generator in $\mathcal{G}$) hits f_n for sufficiently large n, since f_n can be computed by a formula of size $n^{O(1)}$. Hence $\mathcal{G}$ hits VF as claimed. □

Once again, we can improve the stretch of the generator of Lemma 4.5 from 1 to $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$ by applying $n^{1-\varepsilon}$ copies of the generator in parallel.

Theorem 4.6. *Let $\mathbb{F}$ be a field of characteristic zero and let $\varepsilon > 0$ be a fixed constant. Assume that any border formula which computes the $n \times n$ determinant $\det_n(X)$ has size $n^{\omega(1)}$. Then there is an explicit sequence of polynomial maps $\mathcal{G} = (\mathcal{G}_n : \mathbb{F}^{n-n^{1-\varepsilon}} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ such that $\mathcal{G}$ is a VNC^0-computable hitting set generator for* VF*.*

Proof. Let $\hat{\mathcal{G}}$ be the generator of Lemma 4.5. We take $\mathcal{G}_n : \mathbb{F}^{n-n^{1-\varepsilon}} \to \mathbb{F}^n$ to be the concatenation of $n^{1-\varepsilon}$ independent copies of $\hat{\mathcal{G}}_{n^\varepsilon}$. The VNC^0-computability of $\hat{\mathcal{G}}$ implies that $\mathcal{G}$ is VNC^0-computable. The fact that $\mathcal{G}$ hits VF follows from the fact that $\hat{\mathcal{G}}$ hits VF, using a hybrid argument in exactly the same manner as in the proof of Theorem 4.3. □

4.3 Branching Programs

In our final application, we construct a VNC^0-computable generator that hits VBP. To do this, we make use of the fact that VBP is closed under factoring.

Theorem 4.7 ([ST21b]). *Let $\mathbb{F}$ be a field of characteristic zero. Suppose that $f(\overline{x})$ can be computed by an algebraic branching program of size s. Then every irreducible factor of f can be computed by an algebraic branching program of size $s^{O(1)}$.*

The contrapositive of Theorem 4.7 says that if a polynomial f requires branching programs of super-polynomial size, then the same is true for all multiples of f. This allows us to instantiate Corollary 3.8 under the assumption that there is a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VP}$ such that f_n requires branching programs of size $n^{\omega(1)}$.

Lemma 4.8. *Let $\mathbb{F}$ be a field of characteristic zero. Assume there is a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VP}$ such that f_n requires branching programs of size $n^{\omega(1)}$. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes f_n. Let $(\overline{\alpha}^{(n)})_{n\in\mathbb{N}}$ be a sequence of points in $\mathbb{F}^n$ and let $(\beta_n)_{n\in\mathbb{N}}$ be a sequence of field elements. Let $\mathcal{G}_n : \mathbb{F}^{n+s(n)} \to \mathbb{F}^{n+s(n)+1}$ be the local encoding of $\Phi_n(\overline{\alpha}^{(n)}) = \beta_n$. Then $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbb{N}}$ is a VNC^0-computable hitting set generator for VBP.*

Proof. The fact that $\mathcal{G}_n$ is VNC^0-computable follows immediately from Lemma 3.2. To show that $\mathcal{G}_n$ hits VBP, we use Corollary 3.8, Theorem 4.7, and the assumed lower bound on the p-family $(f_n)_{n\in\mathbb{N}}$. By assumption, any branching program that computes $f_n - \beta_n$ must have size $n^{\omega(1)}$. Theorem 4.7 implies that any multiple of $f_n - \beta_n$ likewise requires branching programs of size $t \geqslant n^{\omega(1)}$. It follows from Corollary 3.8 that $\mathcal{G}_n$ hits branching programs of size $\varepsilon\sqrt{t} \geqslant n^{\omega(1)}$ for some constant $\varepsilon > 0$. In particular, for a p-family $(h_n)_{n\in\mathbb{N}} \in \mathrm{VBP}$, the map $\mathcal{G}_q$ (where $q(n) \geqslant n^{\Omega(1)}$ is the index of the n-output generator in $\mathcal{G}$) hits h_n for sufficiently large n, since h_n can be computed by a branching program of size $n^{O(1)}$. Hence $\mathcal{G}$ hits VBP. □

Once again, we can improve the stretch of the generator of Lemma 4.8 from 1 to $n^{1-\varepsilon}$ by applying $n^{1-\varepsilon}$ copies of the generator in parallel. The proof is identical to the proofs of Theorem 4.3 and Theorem 4.6, so we omit the details.

Theorem 4.9. *Let $\mathbb{F}$ be a field of characteristic zero and let $\varepsilon > 0$ be a fixed constant. Assume there is a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VP}$ such that f_n requires branching programs of size $n^{\omega(1)}$. Then there is an explicit sequence of polynomial maps $\mathcal{G} = (\mathcal{G}_n : \mathbb{F}^{n-n^{1-\varepsilon}} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ such that $\mathcal{G}$ is a VNC^0-computable hitting set generator for VBP.*

5 The Ideal Proof System and Cryptographic Generators

We now turn our attention to algebraic proof complexity. An easy observation, already due to Grochow, Kumar, Saks, and Saraf [GKSS17, Section 4.2], shows that a hitting set generator $\mathcal{G}$ for a circuit class $\mathscr{C}$ implies lower bounds against the Geometric $\mathscr{C}$-Ideal Proof System. If the generator is $\mathscr{D}$-computable, then the hard instance for Geometric $\mathscr{C}$-IPS consists of polynomials that are likewise computable in $\mathscr{D}$, which we term $\mathscr{D}$-equations following Definition 2.15. Together with our construction of VNC^0-computable hitting set generators from Section 4, this lets us conclude lower bounds against Geometric $\mathscr{C}$-IPS for $\mathscr{C} \in \{\mathrm{VAC}^0, \mathrm{VF}, \mathrm{VBP}\}$ for systems of VNC^0-equations.

5.1 Geometric IPS Lower Bounds from Hitting Set Generators

We begin with the relationship between hitting set generators and Geometric IPS, which was already observed by Grochow, Kumar, Saks, and Saraf [GKSS17, Section 4.2]. Suppose $\mathcal{G} = (g_1(\overline{y}), \ldots, g_n(\overline{y}))$ is a generator that hits a class of polynomials $\mathscr{C}$. This means that for every nonzero $f \in \mathbb{F}[\overline{x}]$ satisfying $(f \circ \mathcal{G})(\overline{y}) = 0$, we have the non-membership $f \notin \mathscr{C}$. Consider the system of polynomial equations $\mathcal{F} = \{g_1(\overline{y}) = 0, \ldots, g_n(\overline{y}) = 0\}$, and suppose that the system $\mathcal{F}$ is unsatisfiable and can

be refuted by Geometric IPS. Let $r(\overline{x})$ be a Geometric IPS refutation of $\mathcal{F}$. By definition, the polynomial r satisfies the equations

$$r(0,\dots,0) = 1$$
$$r(g_1(\overline{y}),\dots,g_n(\overline{y})) = 0.$$

The polynomials that satisfy these equations are precisely the annihilators of $\mathcal{G}$ whose constant term is 1. Because the refutation r is a nonzero annihilator of $\mathcal{G}$ and $\mathcal{G}$ hits $\mathscr{C}$, we conclude that $r \notin \mathscr{C}$, which is a lower bound on the complexity of r.

Thus, Geometric IPS refutations are simply annihilators with a nonzero constant term. It seems reasonable to conjecture that if there is an easily-computable annihilator, and an annihilator with a nonzero constant term exists, then there is also an easily-computable annihilator with a nonzero constant term. Put more informally, it is not outlandish to think that if we can prove Geometric IPS lower bounds, then we can upgrade the hard system of equations into a hitting set generator. We are unable to prove such a statement. Despite this, we believe that studying Geometric IPS is a meaningful stepping stone towards obtaining hitting set generators. This very paper is evidence for our belief: we first proved lower bounds against Geometric IPS, and only later realized the same construction yields hitting set generators with a minor variation on the proof.

We formalize the preceding argument in the following lemma.

Lemma 5.1 ([GKSS17])**.** *Let $\mathbb{F}$ be a field and let $\mathscr{C}$ and $\mathscr{D}$ be complexity classes. Let $\mathcal{G} = (\mathcal{G}_n : \mathbb{F}^{\ell(n)} \to \mathbb{F}^n)_{n\in\mathbb{N}}$ be a $\mathscr{D}$-computable hitting set generator for $\mathscr{C}$. Then the family of equations $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ given by*

$$\mathcal{F}_n = \{g_{n,1}(\overline{y}) = 0,\dots,g_{n,n}(\overline{y}) = 0\},$$

where $\mathcal{G}_n = (g_{n,1},\dots,g_{n,n})$, is a family of $\mathscr{D}$-computable equations that is hard for Geometric $\mathscr{C}$-IPS.

Proof. Let $(r_n)_{n\in\mathbb{N}}$ be a p-family such that $r_n(\overline{x})$ is a Geometric IPS refutation of $\mathcal{F}_n$. The definition of a Geometric IPS refutation implies

$$r_n(0,\dots,0) = 1$$
$$r_n(g_{n,1}(\overline{y}),\dots,g_{n,n}(\overline{y})) = 0.$$

The first equality implies $r_n(\overline{x}) \neq 0$, while the second implies $r_n \in \mathrm{Ann}(\mathcal{G}_n)$. Because $\mathcal{G}$ hits $\mathscr{C}$, it follows that $r \notin \mathscr{C}$, hence there is no Geometric $\mathscr{C}$-IPS refutation of $\mathcal{F}$. □

We note that in the setting of Lemma 5.1, neither the explicitness nor the stretch of the hitting set generator are of primary concern. The resulting lower bounds are interesting even when the generator is non-explicit and has stretch 1.

5.2 Lower Bounds for Geometric IPS

We now apply Lemma 5.1 to the hitting set generators of Section 4, obtaining systems of VNC^0-equations that are hard for subsystems of Geometric IPS. The lower bounds we obtain here directly correspond to the generator constructions of Section 4: we obtain an unconditional lower bound against Geometric VAC^0-IPS and conditional lower bounds against Geometric VF-IPS and Geometric VBP-IPS.

We stress that because Geometric IPS is an incomplete proof system, it is not enough for our hard instances to be unsatisfiable; we must ensure that Geometric IPS is powerful enough to refute these systems of equations in the first place. For the systems we consider, this property is easy

to verify, as the polynomial that generates the annihilator ideal $\mathrm{Ann}(\mathcal{G})$ is a valid Geometric IPS refutation.

We start with a system of VNC^0-computable equations that are unconditionally hard to refute in Geometric VAC^0-IPS.

Theorem 5.2. *Let $\mathbb{F}$ be a field of characteristic zero. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes the $n \times n$ determinant $\det_n(X)$. Let $(A_n)_{n\in\mathbb{N}}$ be a sequence of $n \times n$ matrices such that $\det(A_n) \neq 0$ for all $n \in \mathbb{N}$. Let $\mathcal{G}_n : \mathbb{F}^{n^2+s(n)} \to \mathbb{F}^{n^2+s(n)+1}$ be the local encoding of $\Phi_n(A_n) = 0$. Let $\mathcal{F}_n$ be the system of equations given by*

$$\mathcal{F}_n = \{g_{n,1}(\overline{y}) = 0, \ldots, g_{n,n^2+s(n)+1}(\overline{y}) = 0\},$$

where $\mathcal{G}_n = (g_{n,1}, \ldots, g_{n,n^2+s(n)+1})$. Then $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ is a sequence of unsatisfiable VNC^0-equations that can be refuted by Geometric VP-IPS, but cannot be refuted by Geometric VAC^0-IPS.

Proof. The fact that $\mathcal{F}_n$ is a system of VNC^0-equations that cannot be refuted by Geometric VAC^0-IPS follows from Lemmas 4.2 and 5.1. To show that $\mathcal{F}_n$ can be refuted by Geometric IPS, let $h_n(\overline{z}) \in \mathrm{Ann}(\mathcal{G}_n)$ be the generator of the ideal $\mathrm{Ann}(\mathcal{G}_n)$. By Proposition 3.6, we know that h_n is given by

$$h_n(\overline{z}) = z_{n^2+s+1} - \det(Z_{1\ldots n^2} + A_n) + g(\overline{z}),$$

where $Z_{1\ldots n^2}$ denotes an $n \times n$ matrix formed from the variables $z_1, \ldots, z_{n^2}$ and $g(\overline{z})$ is a polynomial that satisfies $g(\overline{0}) = 0$. The constant term of h_n is given by

$$h_n(\overline{0}) = -\det(A_n) \neq 0.$$

In particular, the polynomial $\frac{-1}{\det(A_n)}h_n(\overline{z})$ is a Geometric IPS refutation of $\mathcal{F}_n$. The soundness of Geometric IPS implies that $\mathcal{F}_n$ is unsatisfiable. Proposition 3.6 implies that $\frac{-1}{\det(A_n)}h_n(\overline{z})$ can be computed by a circuit of size $O(s) \leqslant n^{O(1)}$, so $(h_n)_{n\in\mathbb{N}} \in \mathrm{VP}$. Hence Geometric VP-IPS can refute $\mathcal{F}$ as claimed. □

Next, we prove a conditional lower bound against Geometric VF-IPS. Our lower bound for Geometric VF-IPS holds assuming a super-polynomial lower bound on the border formula complexity of the determinant. This is the same condition used in Theorem 4.6 to design a VNC^0-computable hitting set generator against VF.

Theorem 5.3. *Let $\mathbb{F}$ be a field of characteristic zero. Assume that any border formula which computes the $n \times n$ determinant $\det_n(X)$ must have size $n^{\omega(1)}$. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes the $n \times n$ determinant $\det_n(X)$. Let $(A_n)_{n\in\mathbb{N}}$ be a sequence of $n \times n$ matrices such that $\det(A_n) \neq 0$ for all $n \in \mathbb{N}$. Let $\mathcal{G}_n : \mathbb{F}^{n^2+s(n)} \to \mathbb{F}^{n^2+s(n)+1}$ be the local encoding of $\Phi_n(A_n) = 0$. Let $\mathcal{F}_n$ be the system of equations given by*

$$\mathcal{F}_n = \{g_{n,1}(\overline{y}) = 0, \ldots, g_{n,n^2+s(n)+1}(\overline{y}) = 0\},$$

where $\mathcal{G}_n = (g_{n,1}, \ldots, g_{n,n^2+s(n)+1})$. Then $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ is a sequence of unsatisfiable VNC^0-equations that can be refuted by Geometric VP-IPS, but cannot be refuted by Geometric VF-IPS.

Proof. The fact that $\mathcal{F}_n$ is a system of VNC^0-equations that cannot be refuted by Geometric VF-IPS follows from the assumed lower bound on the border formula complexity of the determinant together with Lemmas 4.5 and 5.1. This is the same system of equations that appears in Theorem 5.2, so the fact that $\mathcal{F}$ can be refuted by Geometric VP-IPS follows from Theorem 5.2. □

Our final application to Geometric IPS lower bounds is a conditional lower bound against Geometric VBP-IPS. We use the same assumption as in Theorem 4.9, where we constructed a VNC^0-computable generator that hits VBP. That is, we assume there is a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VP}$ such that the branching program complexity of f_n grows super-polynomially in n.

Theorem 5.4. *Let $\mathbb{F}$ be a field of characteristic zero. Assume there is a p-family $(f_n)_{n\in\mathbb{N}} \in \mathrm{VP}$ such that f_n requires branching programs of size $n^{\omega(1)}$. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of arithmetic circuits of size $s(n) \leqslant n^{O(1)}$ such that Φ_n computes f_n. Let $(\overline{\alpha}^{(n)})_{n\in\mathbb{N}}$ be a sequence of points in $\mathbb{F}^n$ and $(\beta_n)_{n\in\mathbb{N}}$ be a sequence of field elements such that $f_n(\overline{\alpha}^{(n)}) \neq \beta_n$ for all n. Let $\mathcal{G}_n : \mathbb{F}^{n+s(n)} \to \mathbb{F}^{n+s(n)+1}$ be the local encoding of $\Phi_n(\overline{\alpha}^{(n)}) = \beta_n$. Let $\mathcal{F}_n$ be the system of equations given by*

$$\mathcal{F}_n = \{g_{n,1}(\overline{y}) = 0, \ldots, g_{n,n+s(n)+1}(\overline{y}) = 0\},$$

where $\mathcal{G}_n = (g_{n,1}, \ldots, g_{n,n+s(n)+1})$. Then $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ is a sequence of unsatisfiable VNC^0-equations that can be refuted by Geometric VP-IPS, but cannot be refuted by Geometric VBP-IPS.

Proof. The fact that $\mathcal{F}_n$ is a system of VNC^0-equations that cannot be refuted by Geometric VBP-IPS follows from Lemmas 4.8 and 5.1. To show that $\mathcal{F}_n$ can be refuted by Geometric IPS, let $h_n(\overline{z}) \in \mathrm{Ann}(\mathcal{G}_n)$ be the generator of $\mathrm{Ann}(\mathcal{G}_n)$. By Proposition 3.6, we know that h_n is given by

$$h(\overline{z}) = z_{n+s+1} - f_n(z_1 + \alpha_1, \ldots, z_n + \alpha_n) + g(\overline{z}) + \beta_n,$$

where $g(\overline{z})$ is a polynomial satisfying $g(\overline{0}) = 0$. The constant term of h_n is given by

$$h_n(\overline{0}) = \beta_n - f_n(\overline{\alpha}) \neq 0,$$

so $\frac{1}{\beta_n - f_n(\overline{\alpha})} h_n(\overline{0}) = 1$. In particular, the polynomial $\frac{1}{\beta_n - f_n(\overline{\alpha})} h_n(\overline{z})$ is a Geometric IPS refutation of $\mathcal{F}_n$. The soundness of Geometric IPS implies that $\mathcal{F}_n$ is unsatisfiable. Moreover, Proposition 3.6 implies that $\frac{1}{\beta_n - f_n(\overline{\alpha})} h_n$ can be computed by a circuit of size $O(s) \leqslant n^{O(1)}$, so $(h_n)_{n\in\mathbb{N}} \in \mathrm{VP}$. Hence Geometric VP-IPS can refute $\mathcal{F}_n$. □

6 Conclusion and Open Problems

In this work, we gave new constructions of VNC^0-computable hitting set generators for fairly strong circuit classes, whose proof of correctness is based on lower bounds for circuit complexity rather than degree. Our constructions followed a general paradigm: from a separation of complexity classes $\mathscr{C} \subsetneq \mathscr{D} \subseteq \mathrm{VP}$, we constructed a VNC^0-computable generator that hits $\mathscr{C}$ by using a local encoding of a p-family in the difference $\mathscr{D} \setminus \mathscr{C}$. The seed length of the generator corresponds to the circuit complexity of the hard p-family in $\mathscr{D} \setminus \mathscr{C}$. This means that we cannot hope to use local encodings to construct generators that hit VP. Because VP is a class of low-degree polynomials, the same argument as in Section 1.2 shows that degree lower bounds for annihilators can be used to construct VNC^0-computable generators for VP. Can one construct VNC^0-computable generators for VP whose correctness is based on circuit complexity, not on degree bounds?

A natural approach to this problem would be to take a known hitting set generator $\mathcal{G}$ for VP and compile it into a different generator $\mathcal{G}'$ that may have worse stretch than $\mathcal{G}$, but improves on the complexity of $\mathcal{G}$ and retains its ability to hit VP. Such a compiler underlies the NC^0-computable one-way functions and pseudorandom generators constructed by Applebaum, Ishai, and Kushilevitz [AIK06]. We do not know how to construct such a compiler in the algebraic setting. One can obtain a new generator $\mathcal{G}'$ by encoding the generator $\mathcal{G}$ by extension variables, similar to how we represent circuits by local encodings, and the resulting $\mathcal{G}'$ is indeed VNC^0-computable. However,

it's not obvious how to show that lower bounds on the annihilators of $\mathcal{G}$ imply lower bounds on annihilators of the new generator $\mathcal{G}'$. It may be the case that by transforming $\mathcal{G}$ to $\mathcal{G}'$, we introduce new annihilating polynomials of lower complexity. Our work shows that this is not possible for generators of the form $\overline{x} \mapsto (\overline{x}, f(\overline{x}))$, but it is less clear how to reason about other base generators $\mathcal{G}$.

For example, if we take $\mathcal{G}$ to be the Kabanets–Impagliazzo generator instantiated with the permanent, can we show that its local encoding $\mathcal{G}'$ hits VP, assuming circuit lower bounds for the permanent? The obvious approach to proving this would be to take an annihilator of the local encoding $\mathcal{G}'$ and obtain from it an annihilator of the base generator $\mathcal{G}$ by substituting a gate variable y_i with the polynomial $f_i(\overline{x})$ computed at the i^{th} gate, just as we did in the proof of Proposition 3.6. If the annihilator of $\mathcal{G}'$ remains nonzero under this substitution, we can conclude a lower bound on its complexity, but it may be the case that the annihilator becomes zero under this substitution, in which case we do not know how to conclude a lower bound on its complexity.

Another natural question in this line of work is to improve on the parameters of our VNC^0-computable generators. We are only able to construct generators with sublinear stretch. Can this be improved to linear or superlinear stretch? Or are there inherent tradeoffs between the stretch, degree, and complexity of a hitting set generator?

In our application to lower bounds for the Ideal Proof System, we were only able to establish lower bounds against the Geometric Ideal Proof System. Can these lower bounds be strengthened to hold for subsystems of the Ideal Proof System, such as VAC^0-IPS? Although polynomial identity testing has a clear and simple relationship to Geometric IPS, it is not obvious if there is a useful application of hitting set generators towards proving lower bounds for the Ideal Proof System.

Finally, constructions of NC^0-computable one-way functions and pseudorandom generators have found numerous applications throughout complexity theory and cryptography. Are there useful applications of VNC^0-computable generators within algebraic complexity, or in complexity theory more broadly?

Acknowledgments

We thank Ramprasad Saptharishi for a simpler formulation and proof of Lemma 3.5, Rafael Oliveira for helpful conversations on Nullstellensatz degree bounds, and Abhibhav Garg for useful feedback that improved the presentation of this work.

References

[AF22] Robert Andrews and Michael A. Forbes. "Ideals, Determinants, and Straightening: Proving and Using Lower Bounds for Polynomial Ideals". In: *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022)*. 2022, pp. 389–402. DOI: 10.1145/3519935.3520025 (cit. on pp. 3, 5, 7, 25, 27).

[AGHT24] Yaroslav Alekseev, Dima Grigoriev, Edward A. Hirsch, and Iddo Tzameret. "Semialgebraic Proofs, IPS Lower Bounds, and the τ-Conjecture: Can a Natural Number be Negative?" In: *SIAM Journal on Computing* 53.3 (2024), pp. 648–700. DOI: 10.1137/20M1374523 (cit. on p. 5).

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. "Cryptography in NC^0". In: *SIAM Journal on Computing* 36.4 (2006), pp. 845–888. DOI: 10.1137/S0097539705446950 (cit. on pp. 2, 8, 31).

[Ald84] A. Alder. "Grenzrang und Grenzkomplexität aus algebraischer und topologischer Sicht". Zürich University, 1984 (cit. on p. 12).

[And20] Robert Andrews. "Algebraic Hardness Versus Randomness in Low Characteristic". In: *35th Computational Complexity Conference (CCC 2020)*. Ed. by Shubhangi Saraf. Vol. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 37:1–37:32. DOI: 10.4230/LIPIcs.CCC.2020.37 (cit. on pp. 2, 25).

[BIK+96] Sam Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiři Sgall. "Proof complexity in algebraic systems and bounded depth Frege systems with modular counting". In: *Computational Complexity* 6 (1996), pp. 256–298. DOI: 10.1007/BF01294258 (cit. on p. 5).

[BIKPP96] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. "Lower bounds on Hilbert's Nullstellensatz and propositional proofs". In: *Proceedings of the London Mathematical Society* 73.3 (1996). Preliminary version in the *35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994)*, pp. 1–26. DOI: 10.1112/plms/s3-73.1.1 (cit. on p. 5).

[Bro87] W. Dale Brownawell. "Bounds for the Degrees in the Nullstellensatz". In: *Annals of Mathematics* 126.3 (1987), pp. 577–591. DOI: 10.2307/1971361 (cit. on p. 6).

[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. 6).

[CKS19] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. "Closure Results for Polynomial Factorization". In: *Theory of Computing* 15.13 (2019). Preliminary version in the *33rd Annual Computational Complexity Conference (CCC 2018)*, pp. 1–34. DOI: 10.4086/toc.2019.v015a013 (cit. on p. 2).

[CKSV22] Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. "Quadratic Lower Bounds for Algebraic Branching Programs and Formulas". In: *Comput. Complex.* 31.2 (2022), p. 8. DOI: 10.1007/S00037-022-00223-8 (cit. on p. 3).

[CT25] Prerona Chatterjee and Anamay Tengse. *Lower Bounds from Succinct Hitting Sets*. 2025. DOI: 10.48550/arXiv.2309.07612 (cit. on p. 2).

[DG24] Pranjal Dutta and Sumanta Ghosh. "Advances in Polynomial Identity Testing". In: *SIGACT News* 55.2 (2024), pp. 53–88. DOI: 10.1145/3674159.3674165 (cit. on p. 2).

[DGW09] Zeev Dvir, Ariel Gabizon, and Avi Wigderson. "Extractors And Rank Extractors For Polynomial Sources". In: *computational complexity* 18.1 (2009), pp. 1–58. DOI: 10.1007/s00037-009-0258-4 (cit. on p. 15).

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. "Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits". In: *SIAM J. Comput.* 39.4 (2009), pp. 1279–1293. DOI: 10.1137/080735850 (cit. on p. 2).

[ER93] Richard Ehrenborg and Gian-Carlo Rota. "Apolarity and Canonical Forms for Homogeneous Polynomials". In: *European Journal of Combinatorics* 14.3 (1993), pp. 157–181. DOI: https://doi.org/10.1006/eujc.1993.1022 (cit. on p. 15).

[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. 6).

[FKS16] Michael A. Forbes, Mrinal Kumar, and Ramprasad Saptharishi. "Functional Lower Bounds for Arithmetic Circuits and Connections to Boolean Circuit Complexity". In: *Proceedings of the 31st Annual Computational Complexity Conference (CCC 2016)*. Ed. by Ran Raz. Vol. 50. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016, 33:1–33:19. DOI: 10.4230/LIPIcs.CCC.2016.33 (cit. on p. 5).

[FLST24] Hervé Fournier, Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. "On the Power of Homogeneous Algebraic Formulas". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*. STOC 2024. Vancouver, BC, Canada: Association for Computing Machinery, 2024, pp. 141–151. DOI: 10.1145/3618260.3649760 (cit. on p. 7).

[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. DOI: 10.4230/LIPIcs.CCC.2024.31 (cit. on p. 7).

[FSTW21] Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. "Proof Complexity Lower Bounds from Algebraic Circuit Complexity". In: *Theory of Computing* 17.10 (2021), pp. 1–88. DOI: 10.4086/toc.2021.v017a010. Preliminary version in the *31st Annual Computational Complexity Conference (CCC 2016)* (cit. on p. 5).

[GHT22] Nashlen Govindasamy, Tuomas Hakoniemi, and Iddo Tzameret. "Simple Hard Instances for Low-Depth Algebraic Proofs". In: *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2022)*. 2022, pp. 188–199. DOI: 10.1109/FOCS54457.2022.00025 (cit. on p. 5).

[GKSS17] Joshua A. Grochow, Mrinal Kumar, Michael Saks, and Shubhangi Saraf. *Towards an algebraic natural proofs barrier via polynomial identity testing*. 2017. DOI: 10.48550/arXiv.1701.01717 (cit. on pp. 8, 28, 29).

[GKSS22] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. "Derandomization from Algebraic Hardness". In: *SIAM Journal on Computing* 51.2 (2022), pp. 315–335. DOI: 10.1137/20M1347395 (cit. on p. 2).

[GP18] Joshua A. Grochow and Toniann Pitassi. "Circuit Complexity, Proof Complexity, and Polynomial Identity Testing: The Ideal Proof System". In: *J. ACM* 65.6 (Nov. 2018), 37:1–37:59. DOI: 10.1145/3230742 (cit. on pp. 5–8, 13, 14).

[Gro23] Joshua A. Grochow. *Polynomial Identity Testing and the Ideal Proof System: PIT is in NP if and only if IPS can be p-simulated by a Cook-Reckhow proof system*. 2023. DOI: 10.48550/arXiv.2306.02184 (cit. on p. 9).

[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. 6).

[HLT24] Tuomas Hakoniemi, Nutan Limaye, and Iddo Tzameret. "Functional Lower Bounds in Algebraic Proofs: Symmetry, Lifting, and Barriers". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC 2024)*. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1396–1404. DOI: 10.1145/3618260.3649616 (cit. on pp. 5, 6).

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. "Lower bounds for the polynomial calculus and the Gröbner basis algorithm". In: *Computational Complexity* 8 (1999), pp. 127–144. DOI: 10.1007/s000370050024 (cit. on p. 5).

[Jac41] C. G. J. Jacobi. "De Determinantibus functionalibus". In: *Journal für die reine und angewandte Mathematik* 1841.22 (1841), pp. 319–359. DOI: 10.1515/crll.1841.22.319 (cit. on p. 15).

[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. 6).

[Kal87] Erich Kaltofen. "Single-Factor Hensel Lifting and its Application to the Straight-Line Complexity of Certain Polynomials". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. 1987, pp. 443–452. DOI: 10.1145/28395.28443 (cit. on p. 9).

[Kay09] Neeraj Kayal. "The Complexity of the Annihilating Polynomial". In: *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC 2009)*. 2009, pp. 184–193. DOI: 10.1109/CCC.2009.37 (cit. on pp. 3, 5, 19).

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Computational Complexity* 13.1-2 (2004), pp. 1–46. DOI: 10.1007/s00037-004-0182-6 (cit. on p. 2).

[Kol88] János Kollár. "Sharp effective Nullstellensatz". In: *J. Amer. Math. Soc.* 1.4 (1988), pp. 963–975. DOI: 10.2307/1990996 (cit. on p. 6).

[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. 6).

[KS22] Deepanshu Kush and Shubhangi Saraf. "Improved Low-Depth Set-Multilinear Circuit Lower Bounds". In: *37th Computational Complexity Conference (CCC 2022)*. Ed. by Shachar Lovett. Vol. 234. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 38:1–38:16. DOI: 10.4230/LIPIcs.CCC.2022.38 (cit. on p. 7).

[KS23] Deepanshu Kush and Shubhangi Saraf. "Near-Optimal Set-Multilinear Formula Lower Bounds". In: *38th Computational Complexity Conference (CCC 2023)*. Ed. by Amnon Ta-Shma. Vol. 264. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 15:1–15:33. DOI: 10.4230/LIPIcs.CCC.2023.15 (cit. on p. 7).

[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 pp. 2, 3, 7, 25).

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *J. Comput. System Sci.* 49.2 (1994), pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1 (cit. on p. 2).

[Raz98] Alexander A. Razborov. "Lower bounds for the polynomial calculus". In: *Computational Complexity* 7 (1998), pp. 291–324. DOI: 10.1007/s000370050013 (cit. on p. 5).

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. "On the Range Avoidance Problem for Circuits". In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 640–650. DOI: 10.1109/FOCS54457.2022.00067 (cit. on p. 2).

[Sap] Ramprasad Saptharishi. *A survey of lower bounds in arithmetic circuit complexity*. https://github.com/dasarpmar/lowerbounds-survey (cit. on p. 10).

[Sax09] Nitin Saxena. "Progress on Polynomial Identity Testing". In: *Bulletin of the EATCS* 99 (2009), pp. 49–79 (cit. on p. 2).

[Sax14] Nitin Saxena. "Progress on Polynomial Identity Testing II". In: *Proceedings of the Workshop celebrating Somenath Biswas' 60th Birthday*. 2014, pp. 131–146 (cit. on p. 2).

[Sch80] Jacob T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* 27.4 (1980), pp. 701–717. DOI: 10.1145/322217.322225 (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. 6).

[ST21a] Rahul Santhanam and Iddo Tzameret. "Iterated Lower Bound Formulas: A Diagonalization-Based Approach to Proof Complexity". In: *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 234–247. DOI: 10.1145/3406325.3451010 (cit. on p. 5).

[ST21b] Amit Sinhababu and Thomas Thierauf. "Factorization of Polynomials Given by Arithmetic Branching Programs". In: *Computational Complexity* 30.15 (2021). DOI: 10.1007/s00037-021-00215-0 (cit. on pp. 2, 9, 28).

[SY10] Amir Shpilka and Amir Yehudayoff. "Arithmetic Circuits: A survey of recent results and open questions". In: *Foundations and Trends in Theoretical Computer Science* 5.3-4 (2010), pp. 207–388. DOI: 10.1561/0400000039 (cit. on p. 10).

[TLS22] Sébastien Tavenas, Nutan Limaye, and Srikanth Srinivasan. "Set-multilinear and non-commutative formula lower bounds for iterated matrix multiplication". In: *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022)*. STOC 2022. Rome, Italy: Association for Computing Machinery, 2022, pp. 416–425. DOI: 10.1145/3519935.3520044 (cit. on p. 7).

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge University Press, 2013 (cit. on p. 16).

[Zip79] Richard Zippel. "Probabilistic algorithms for sparse polynomials". In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation, EUROSAM 1979*. 1979, pp. 216–226. DOI: 10.1007/3-540-09519-5_73 (cit. on p. 1).

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il