THEORY OF COMPUTING, Volume 19 (12), 2023, pp. 1–30
www.theoryofcomputing.org

Near-Optimal Bootstrapping of Hitting Sets for Algebraic Models

Mrinal Kumar* Ramprasad Saptharishi† Anamay Tengse‡

Received April 16, 2019; Revised August 5, 2021; Published December 31, 2023

Abstract. The Polynomial Identity Lemma (also called the "Schwartz–Zippel lemma") states that any nonzero polynomial $f(x_1, \ldots, x_n)$ of degree at most s will evaluate to a nonzero value at some point on any grid $S^n \subseteq \mathbb{F}^n$ with $|S| > s$. Thus, there is an explicit *hitting set* for all n-variate degree-s, size-s algebraic circuits of size $(s+1)^n$.

In this paper, we prove the following results:

A preliminary version of this paper appeared in the Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019) [20].

*Tata Institute of Fundamental Research, Mumbai, India. A part of this work was done during postdoctoral stays at Simons Institute for the Theory of Computing, Berkeley, USA, Center for Mathematical Sciences and Applications at Harvard, and while visiting TIFR, Mumbai.

†Tata Institute of Fundamental Research, Mumbai, India. Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500

‡Reichman University, Herzliya. A major part of this work was done as a student at TIFR, Mumbai, when the author was supported by a fellowship of the DAE, Govt. of India.

ACM Classification: F.2.1, F.1.3

AMS Classification: 68W30, 68Q87, 68Q06

Key words and phrases: algebraic circuits, polynomial identity testing, derandomization, hardness-randomness, bootstrapping

 DOI: 10.4086/toc.2023.v019a012

- Let $\epsilon > 0$ be a constant. For a sufficiently large constant n, and all $s > n$, if we have an explicit hitting set of size $(s+1)^{n-\epsilon}$ for the class of n-variate degree-s polynomials that are computable by algebraic circuits of size s, then for all large s, we have an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for s-variate circuits of degree s and size s.

  That is, if we can obtain a barely non-trivial exponent (a factor-$s^{\Omega(1)}$ improvement) compared to the trivial $(s+1)^n$-size hitting set even for constant-variate circuits, we can get an almost complete derandomization of PIT.

- The above result holds when "circuits" are replaced by "formulas" or "algebraic branching programs."

This extends a recent surprising result of Agrawal, Ghosh and Saxena (STOC 2018, PNAS 2019) who proved the same conclusion for the class of algebraic circuits, if the hypothesis provided a hitting set of size at most $\left(s^{n^{0.5-\delta}}\right)$ (where $\delta > 0$ is any constant). Hence, our work significantly weakens the hypothesis of Agrawal, Ghosh and Saxena to only require a slightly non-trivial saving over the trivial hitting set, and also presents the first such result for algebraic formulas.

1 Introduction

Multivariate polynomials are the primary protagonists in the field of algebraic complexity and algebraic circuits form a natural robust model of computation for multivariate polynomials. For completeness, we now define algebraic circuits : an algebraic circuit is a directed acyclic graph with internal gates labeled by + (addition) and × (multiplication), and with leaves labeled by either variables or field constants; computation flows in the natural way.

In the field of algebraic complexity, much of the focus has been restricted to studying n-variate polynomials whose degree is bounded by a polynomial function in n, and such polynomials are called *low-degree polynomials*. This restriction has several *a priori* and *a posteriori* motivations, and excellent discussions of this can be seen in the thesis of Forbes [12, Section 3.2] and Grochow's answer [13] on `cstheory.SE`. The central question in algebraic complexity is to find a family of low-degree polynomials that requires large algebraic circuits to compute it. Despite having made substantial progress in various subclasses of algebraic circuits (see, e. g., the surveys [30, 26]), the current best lower bound for general algebraic circuits is merely an $\Omega(n \log d)$ lower bound of Baur and Strassen [5].

An interesting approach towards proving lower bounds for algebraic circuits is via showing good *upper bounds* for the *algorithmic* task of *polynomial identity testing.* Our results in this paper deal with this problem, and we elaborate on this now.

1.1 Polynomial Identity Testing

Polynomial identity testing (PIT[1]) is the algorithmic task of checking if a given algebraic circuit C of size s computes the identically zero polynomial. As discussed earlier, although a circuit of size s can compute a polynomial of degree 2^s, this question typically deals only with circuits whose *formal degree*[2] is bounded by the size of the circuit.

PIT is an important algorithmic question in its own right, and many classical results such as the primality testing algorithm [3], $\mathsf{IP} = \mathsf{PSPACE}$ [21, 29], algorithms for graph matching [22, 11, 32] all have polynomial identity tests at their core.

This algorithmic question has two flavors: whitebox PIT and blackbox PIT. Whitebox polynomial identity tests consist of algorithms that can inspect the circuit (that is, look at the underlying gate connections etc.) to decide whether the circuit computes the zero polynomial or not. A stronger algorithm is a *blackbox polynomial identity test* where the algorithm is only provided basic parameters of the circuit (such as its size, the number of variables, a bound on the formal degree) and only has evaluation access to the circuit C. Hence, a blackbox polynomial identity test for a class $\mathcal{C}$ of circuits is essentially just a list of evaluation points $H \subseteq \mathbb{F}^n$ such that every nonzero circuit $C \in \mathcal{C}$ is guaranteed to have some $\mathbf{a} \in H$ such that $C(\mathbf{a}) \neq 0$. Such sets of points are also called *hitting sets* for $\mathcal{C}$. Therefore, the running time of a blackbox PIT algorithm is given by the size of the *hitting set*, the time taken to generate it given the parameters of the circuit, and the time taken to evaluate the circuit on these points. We shall say that a hitting set H is *explicit* if there is an algorithm that, given the parameters n, d, s, outputs the set H in time that is polynomial in the *bit complexity*[3] of H.

The classical Polynomial Identity Lemma[4] [25, 9, 33, 28] states the following.

Lemma 1.1 (Polynomial Identity Lemma)**.** *Any nonzero polynomial $f(x_1, \dots, x_n)$ of degree at most d will evaluate to a nonzero value at a randomly chosen point from a finite grid $S^n \subseteq \mathbb{F}^n$ with probability at least $1 - \frac{d}{|S|}$.*

This automatically yields a *randomized* polynomial-time blackbox PIT algorithm, and also an explicit hitting set of size $(d+1)^n$, for the class of n-variate formal degree-d polynomials. Furthermore, a simple counting/dimension argument also says that there exist (non-explicit) $\mathrm{poly}(s)$-size hitting sets for the class of polynomials computed by size-s algebraic circuits. The major open question is to find a better *deterministic* algorithm for this problem, and the task of constructing deterministic PIT algorithms is intimately connected with the question of proving explicit lower bounds for algebraic circuits.

Heintz and Schnorr [15], and Agrawal [1] observed that given an explicit hitting set for size-s circuits, any nonzero polynomial that is designed to vanish on every point of the hitting set

[1]We use the abbreviation PIT for both the noun 'polynomial identity test' and gerund/adjective 'polynomial identity testing'. The case would be clear from context.

[2]This is defined inductively by setting the formal degree of leaves as 1, and taking the sum at every multiplication gate and the max at every sum gate.

[3]Throughout the paper, we will only talk about the sizes of the hitting sets as their bit complexities (i. e., the total bit-lengths) are always bounded by a polynomial in their respective sizes. We provide more details about this analysis in subsection 4.1.

[4]A discussion of the history of this result can be found in [6].

cannot be computable by size-s circuits. By tailoring the number of variables and degree of the polynomial in this observation, they showed that polynomial-time blackbox PITs yield an E-computable family $\{f_n\}$ of n-variate multilinear polynomials that require $2^{\Omega(n)}$-size circuits. This connection between PIT and lower bounds was strengthened further by Kabanets and Impagliazzo [18] who showed that explicit families of hard functions can be used to give non-trivial derandomizations for PIT. Thus, the question of proving explicit lower bounds and the task of finding upper bounds for PIT are essentially two sides of the same coin.

1.2 Bootstrapping

A recent result of Agrawal, Ghosh and Saxena [2] showed, among other things, the following surprising result: blackbox PIT algorithms for size-s and n-variate circuits with running time as bad as $\left(s^{n^{0.5-\delta}}\right)$, where $\delta > 0$ is a constant, can be used to construct blackbox PIT algorithms for size-s circuits with running time $s^{\exp(\exp(O(\log^* s)))}$. Note that $\log^* n$ refers to the smallest i such that the i-th iterated logarithm $\log^{\circ i}(n)$ is at most 1. This shows that good-enough derandomizations of PIT would be sufficient to get a nearly complete derandomization. Their proof uses a novel *bootstrapping* technique where they use the connection between hardness and derandomization repeatedly so that by starting with a weak hitting set we can obtain better and better hitting sets.

One of the open questions of Agrawal, Ghosh and Saxena [2] was whether the hypothesis can be strengthened to a *barely* non-trivial derandomization. That is, suppose we have a blackbox PIT algorithm, for the class of n-variate, size-s circuits that runs in time $s^{o(n)}$, can we use this to get a nearly complete derandomization? Note that we have a trivial $(s+1)^n \cdot \mathrm{poly}(s)$ algorithm from the Polynomial Identity Lemma (Lemma 1.1). Our main result is an affirmative answer to this question in a very strong sense. Furthermore, our result holds for *typical* subclasses that are reasonably well-behaved under composition. Formally, we prove the following theorem.

Theorem 1.2 (Bootstrapping PIT for algebraic formulas, branching programs and circuits)**.** *Let $\epsilon > 0$ and $n \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $s^{n-\epsilon}$ for all degree-s, size-s algebraic formulas (or algebraic branching programs or circuits) over n variables. Then, for all large s, there is an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for the class of degree-s, size-s algebraic formulas (or algebraic branching programs or circuits, respectively) over s variables.*

Remark 1.3. While the hypothesis of Theorem 1.2 above assumes that n is a constant, qualitatively similar results continue to hold even when n is slowly growing with s. For simplicity of notation, we work with constant n throughout the paper and discuss the extension to growing n in more detail in subsection 4.2.

Note that $(s+1)^{n-\epsilon} = s^{n-\epsilon} \cdot \left(1+\frac{1}{s}\right)^{n-\epsilon} < \mathrm{e} \cdot s^{n-\epsilon} < s^{n-\epsilon'}$ for some other constant $\epsilon' > 0$ since s is large enough. Hence, for this theorem, there is no qualitative difference if the hitting set had size $(s+1)^{n-\epsilon}$ instead of $s^{n-\epsilon}$. We also note that as far as we understand, such a statement for classes such as algebraic formulas, even with the stronger hypothesis of there being an $s^{O(n^{(1/2)-\epsilon})}$, did not follow from the results in [2]. We elaborate more on this, and the differences between our proof and theirs in the next subsection.

An interesting, albeit simple corollary of the above result is the following statement.

Corollary 1.4 (From slightly non-trivial PIT to lower bounds)**.** *Let $\epsilon > 0$ and $k \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $(s^{k-\epsilon})$ for all degree-s, size-s algebraic formulas (or algebraic branching programs or circuits) over k variables. Then, for every function $d : \mathbb{N} \to \mathbb{N}$, there is a polynomial family $\{f_n\}$, where f_n is an n-variate of degree $d(n)$, and for every large enough n, f_n cannot be computed by algebraic formulas or algebraic branching programs or circuits of size smaller than $\binom{n+d}{d}^{\frac{1}{\exp(\exp(O(\log^* nd)))}}$. Moreover, there is an algorithm which when given as input an n-variate monomial of degree d, outputs its coefficient in f_n in deterministic time $\binom{n+d}{d}$.*

Thus, a slightly non-trivial blackbox PIT algorithm leads to hard families with near optimal hardness (as any n-variate polynomial of total degree d trivially has a circuit of size $\binom{n+d}{d}$). In a recent result concerning *non-commutative* algebraic circuits, Carmosino, Impagliazzo, Lovett and Mihajlin [8] showed that given an explicit polynomial family of constant degree which requires superlinear size non-commutative circuits, one can obtain explicit polynomial families of exponential hardness. Besides the obvious differences in the statements, one important point to note is that the notions of explicitness in the conclusions of the two statements are different from each other. In [8], the final exponentially hard polynomial family is in VNP provided the initial polynomial family is also in VNP. On the other hand, for our result, we can say that the hard polynomial family obtained in the conclusion is explicit in the sense that its coefficients are computable in deterministic time $\binom{n+d}{d}$. Another difference between Corollary 1.4 and the main result of [8] is in the hypothesis. From a non-trivial hitting set, we can obtain a large class of lower bounds by varying parameters appropriately (see Theorem 1.5), however the main result of [8] starts with a lower bound for a single family. In that regard, our hypothesis appears to be much stronger and slightly non-standard. We discuss this issue in some detail at the end of the next section.

In another relevant result, Jansen and Santhanam [17] showed that marginal improvements to known hitting set constructions imply lower bounds for the permanent polynomial. In particular, they show that a "sufficiently succinct" hitting set of size d, for univariates of degree d that have *constant-free* algebraic circuits of small size, would imply that the permanent polynomial requires superpolynomial-size *constant-free* algebraic circuits. Note that even though their hypothesis needs a much weaker improvement in the size of the hitting set when compared to ours, the hitting set is additionally required to be "succinct"[5], which makes it difficult to compare the two hypotheses.

1.3 Proof overview

The basic intuition for the proofs in this paper, and as per our understanding also for the proofs of the results in the work of Agrawal, Ghosh and Saxena [2], comes from the results of Kabanets

[5]They require their hitting sets to be encoded by uniform TC^0 circuits of appropriately small size. See [17] for details.

and Impagliazzo [18], and those of Heintz and Schnorr [15] and Agrawal [1]. We start by informally stating these results.

Theorem 1.5 (Informal, Heintz and Schnorr [15], Agrawal [1])**.** *Let $H(n, d, s)$ be an explicit hitting set for circuits of size s, degree d in n variables. Then, for every $k \leq n$ and d' such that $d'k \leq d$ and $(d'+1)^k > |H(n, d, s)|$, there is a nonzero polynomial in k variables and individual degree d' that vanishes on the hitting set $H(n, d, s)$, and hence cannot be computed by a circuit of size s.*

In a nutshell, given an explicit hitting set, we can obtain hard polynomials. In fact, playing around with the parameters d' and $k \leq n$, we can get a hard polynomial in k variables, of degree kd' for all k, d' satisfying $d'k < d$ and $(d'+1)^k > |H(n, d, s)|$.

We now state a result of Kabanets and Impagliazzo [18] that shows that hardness can lead to derandomization.

Theorem 1.6 (Informal, Kabanets and Impagliazzo [18])**.** *A superpolynomial lower bound for algebraic circuits for an explicit family of polynomials implies a deterministic blackbox PIT algorithm for all algebraic circuits in n variables and degree d of size* $\mathrm{poly}(n)$ *that runs in time $d^{O(n^{\epsilon})}$ for every $\epsilon > 0$.*

Now, we move on to the main ideas in our proof. Suppose we have non-trivial hitting sets for size-s, degree $d \leq s$ circuits on n variables. The goal is to obtain a blackbox PIT for circuits of size s, degree s on s variables with a much better dependence on the number of variables.

Observe that if the number of variables was much much smaller than s, say at most a constant, then the hitting set in the hypothesis has a polynomial dependence on s, and we are done. We will proceed by presenting *variable reductions* to eventually reach this stage. With this in mind, the hitting sets for s-variate circuits in the conclusion of Theorem 1.2 are designed iteratively starting from hitting sets for circuits with very few variables. In each iteration, we start with a hitting set for size-s, degree $d \leq s$ circuits on n variables with some dependence on n and obtain a hitting set for size-s, degree $d \leq s$ circuits on $m = 2^{n^{\delta}}$ variables (for some $\delta > 0$), that has a *much* better dependence on m. Then, we repeat this process till the number of variables increases up to s, which takes $O(\log^* s)$ iterations. We now briefly outline the steps in each such iteration.

- **Obtaining a family of hard polynomials :** The first step is to obtain a family of explicit hard polynomials from the given hitting sets. This step is done via Theorem 1.5, which simply uses interpolation to find a nonzero polynomial Q in k variables and degree d that vanishes on the hitting set for size-s', degree-d' circuits on n variables, for some s', d' to be chosen appropriately.

- **Variable reduction using Q :** Next we take a combinatorial design (see Definition 2.6) $\{S_1, S_2, \ldots, S_m\}$, where each S_i is a subset of size k of a universe of size $\ell = \mathrm{poly}(k)$, and $|S_i \cap S_j| \ll k$. Consider the map $\Gamma : \mathbb{F}[x_1, x_2, \ldots, x_m] \to \mathbb{F}[y_1, y_2, \ldots, y_\ell]$ given by the substitution $\Gamma(C(x_1, x_2, \ldots, x_m)) = C\left(Q(\mathbf{y}\mid_{S_1}), Q(\mathbf{y}\mid_{S_2}), \ldots, Q(\mathbf{y}\mid_{S_m})\right)$. As Kabanets and Impagliazzo show in the proof of Theorem 1.6, Γ preserves the nonzeroness of all algebraic circuits of size s on m variables, provided Q is hard enough.

We remark that our final argument for this part is slightly simpler than that of Kabanets and Impagliazzo, and hence our results also hold for algebraic formulas. In particular, we do not need Kaltofen's seminal result that algebraic circuits are closed under polynomial factorization, whereas the proof of Kabanets and Impagliazzo crucially uses Kaltofen's result [19]. This comes from the simple, yet effective, observation that if Q vanishes on some hitting set, then so does any multiple of Q. This allows us to use the hardness of *low-degree* multiples of Q, and so, we do not need any complexity guarantees on factors of polynomials.

It is worth mentioning that the result of Kabanets and Impagliazzo [18] is the algebraic analogue of the famous result of Nisan and Wigderson [24] from the boolean world. However, the algebraic setting allows us to construct polynomials with hardness that is *super-exponential* in the number of variables. This is a key reason for bootstrapping to work, and we shall elaborate a bit more on this later in this section.

- **Blackbox PIT for m-variate circuits of size s and degree s :** We now take the hitting set given by the hypothesis for the circuit $\Gamma(C)$ (invoked with appropriate size and degree parameters) and evaluate $\Gamma(C)$ on this set. From the discussion so far, we know that if C is nonzero, then $\Gamma(C)$ cannot be identically zero, and hence it must evaluate to a nonzero value at some point on this set. The number of variables in $\Gamma(C)$ is at most $\ell = \operatorname{poly}\log m$, whereas its size turns out to be *not too much* larger than s. Hence, the size of the hitting set for C obtained via this argument turns out to have a better dependence on the number of variables m than the hitting set in the hypothesis.

To prove Corollary 1.4, we let $t(n) = \exp(\exp(O(\log^* n)))$. Now, we invoke the the conclusion of Theorem 1.2 with $s = \binom{n+d}{d}^{\frac{1}{10t(n)}}$. Thus, we get an explicit hitting set H of size $\binom{n+d}{d}^{\frac{1}{10}}$ for n-variate circuits of size s and degree d. We now use Theorem 1.5 to get a nonzero polynomial of degree d and n which vanishes on the set H and hence cannot be computed by circuits of size at most s. We skip the rest of the details.

Why does bootstrapping work? As far as we understand, the primary reason that makes such bootstrapping results feasible is the following observation from the results of Heintz and Schnorr, and Agrawal [15, 1]: Given a single hitting set, we can obtain a *family* of lower bounds by varying the degree and the number of variables in the interpolating polynomial. It turns out that in the result of Kabanets and Impagliazzo [18] that converts a hard polynomial P into a hitting set, the proof of this conversion has different sensitivities to the degree of P and the number of variables it depends on. The combination of both these facts allows us to start with a moderately non-trivial hitting set, obtaining a hard polynomial from it of the *right* degree and number of variables, and use that to obtain a hitting set which is significantly better than what we started with. In particular, by choosing a degree that is polynomial in the hitting set size, say $|H|^{0.1}$, we can obtain a *constant-variate* polynomial that vanishes on H, for any H. This is impossible in the boolean world where the best hardness one could hope for is exponential in the number of variables. This, in our opinion, is a high level picture of why bootstrapping works in the algebraic world.

Similarities and differences with the proof of Agrawal, Ghosh and Saxena [2]. The high level outline of our proof is essentially the same as that in [2]. However, there are some differences that make our final arguments shorter, simpler and more robust than those of Agrawal, Ghosh and Saxena thus leading to a stronger and near optimal bootstrapping statement in Theorem 1.2. Moreover, as we already alluded to, our proof extends to formulas and algebraic branching programs as well, whereas, to the best of our understanding, the proofs in [2] do not. We now elaborate on the differences.

One of the main differences between the proofs in this paper and those in [2] is in the use of the result of Kabanets and Impagliazzo [18]. Agrawal, Ghosh and Saxena use this result as a blackbox to get deterministic PIT using hard polynomials. The result of Kabanets and Impagliazzo [18] requires a result of Kaltofen, which shows that low-degree algebraic circuits are closed under polynomial factorization. That is, if a degree-d, n-variate polynomial P has a circuit of size at most s, then any factor of P has a circuit of size at most $(snd)^e$ for a constant e. Such a closure result is not known to be true for formulas[6], and hence the results in [2] do not seem to extend to these settings. Also, the removal of any dependence on the "factorization exponent" e is a key ingredient in our proof as it allows us to start with a hypothesis of a barely non-trivial hitting set. To see this more clearly, suppose we wish to start with a hypothesis that gives hitting sets of size $s^{g(n)}$ for n-variate circuits of size and degree s. It is then not too difficult to infer from Lemma 3.2 that for any proof of "variable reduction" using the *hybrid argument* of Kabanets and Impagliazzo [18], we require $g(n)$ to satisfy the relation $e \cdot g(n) \leq n$. This would mean that $s^{g(n)}$ can never be something like $s^{n-\epsilon}$ i. e., a poly(s) factor better than the trivial s^n.

The other main difference between our proof and that in [2] is rather technical but we try to briefly describe it. This is in the choice of combinatorial designs. The designs used in this paper are based on the standard Reed-Solomon code and they yield larger set families than the designs used by [2]. However, even without these improved design parameters, our proof can be used to provide the same conclusion when starting off with a hitting set of size s^{n^δ}, instead of the hypothesis of Theorem 1.2[7].

Also, their proof is quite involved and we are unsure if there are other constraints in their proof that force such choices of parameters. Our proof, though along almost exactly the same lines, appears to be more transparent and more malleable with respect to the choice of parameters.

The strength of the hypothesis. The hypothesis of Theorem 1.2 and also those of the results in the work of Agrawal, Ghosh and Saxena [2] is that we have a non-trivial explicit hitting set for algebraic circuits of size s, degree d on n variables where d and s could be arbitrarily large as functions of n. This seems like an extremely strong assumption, and also slightly non-standard

[6]Even for algebraic branching programs such a result was shown only recently (after our work) by Sinhababu and Thierauf [31].

[7]Even though n, ϵ and δ are constants, n^δ and $(n-\epsilon)$ are qualitatively different, as ϵ is independent of n.

in the following sense. In a typical setting in algebraic complexity, we are interested in PIT for size-s, degree-d circuits on n variables where d and s are polynomially bounded in the number of variables n. A natural open problem here, which would be a more satisfying statement to have, would be to show that one can weaken the hypothesis in Theorem 1.2 to only hold for circuits whose degree and size are both polynomially bounded in n. It is not clear to us if such a result can be obtained using the current proof techniques, or is even true.

Having noted that our hypothesis is very strong, and perhaps even slightly unnatural with respect to the usual choice of parameters in the algebraic setting, we remark that our hypothesis does in fact follow from the assumptions that the Permanent is hard for Boolean circuits, and the Generalized Riemann Hypothesis (GRH). The proof is essentially the same as that of Corollary 1 in the work of Jansen and Santhanam [17]. The only difference is that while Jansen and Santhanam show that there are non-trivial explicit[8] hitting sets for univariate polynomials with small circuits assuming the hardness of Permanent for Boolean circuits and the GRH, here we have to work with circuits computing multivariate polynomials. At a high level, the proof in [17] proceeds by constructing a pseudorandom generator for Boolean circuits of appropriate size assuming the hardness of permanent for Boolean circuits. Then, the set of binary strings in the output of this generator is interpreted in a natural way as an integer. This gives us a small set of integer points, which can be constructed deterministically. Then they argue that there is no constant free algebraic circuit of small size which vanishes on all these integer points. The proof of this step is by contradiction, where they assume the existence of such a constant free algebraic circuit to construct a Boolean circuit of small size which is not fooled by the aforementioned Boolean pseudorandom generator. For algebraic circuits which are not constant free and are allowed to use arbitrary field constants and hence cannot be efficiently simulated by a Boolean circuit, they assume the GRH to reduce to the case of constant free circuits in a fairly standard way. For our setting, we interpret the output of the Boolean pseudorandom generator as not just a single integer point, but a k tuple of integers points. These set of points in $\mathbb{Z}^k$ form our candidate hitting set. The rest of the proof carries over without any changes. We refer the interested reader to [17] for further details.

Remark 1.7. Throughout the paper, we shall assume that there are suitable $\lfloor\cdot\rfloor$'s or $\lceil\cdot\rceil$'s if necessary so that certain parameters chosen are integers. We avoid writing this purely for the sake of readability.

All results in this paper continue to hold for the underlying model of algebraic formulas, algebraic branching programs or algebraic circuits. In fact, the results also extend to the model of *border* of algebraic formulas, algebraic branching programs or algebraic circuits i. e.if there is a slightly non-trivial hitting set for polynomials in the border of these classes, then our main theorem gives a highly non-trivial explicit hitting set for these polynomials. Since our proofs extend as it is to this setting with essentially no changes, we skip the details for this part, and confine our discussions in the rest of the paper to just standard algebraic formulas.

[8]In fact their notion of explicitness is stronger than ours.

2 Preliminaries

Notation

- For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.
- We use boldface letters such as $\mathbf{x}_{[n]}$ to denote a set $\{x_1, \ldots, x_n\}$. We drop the subscript whenever the number of elements is clear or irrelevant in the context.
- For a polynomial $f(x_1, \ldots, x_n)$, we shall say its *individual degree* is at most k to mean that the exponent of any of the x_i's in any monomial is at most k.

We now define some standard notions we work with, and state some of the known results that we use in this paper.

2.1 Algebraic models of computation

Throughout the paper we would be dealing with some standard algebraic models and we define them formally for completeness.

Definition 2.1 (Algebraic branching programs (ABPs))**.** An *algebraic branching program* in variables $\{x_1, x_2, \ldots, x_n\}$ over a field $\mathbb{F}$ is a directed acyclic graph with a designated *starting vertex* s with in-degree zero, a designated *end vertex* t with out-degree zero, and the edge between any two vertices labeled by an affine form from $\mathbb{F}[x_1, x_2, \ldots, x_n]$. The polynomial computed by the ABP is the sum of all weighted paths from s to t, where the weight of a directed path in an ABP is the product of labels of the edges in the path.

The size of an ABP is defined as the number of edges in the underlying graph.

Definition 2.2 (Algebraic formulas)**.** An algebraic circuit is said to be a *formula* if the underlying graph is a tree. The size of a formula is defined as the number of leaves.

The notation $C(n, d, s)$ will be used to denote the class of n-variate[9] polynomials of degree at most d that are computable by formulas of size at most s.

We will use the following folklore algorithm for computing univariate polynomials, often attributed to Horner[10]. We also include a proof for completeness.

Proposition 2.3 (Horner rule)**.** *Let $P(x) = \sum_{i=0}^{d} p_i x^i$ be a univariate polynomial of degree d over any field $\mathbb{F}$. Then, P can be computed by an algebraic formula of size $2d + 1$.*

Proof. Follows from the fact that $P(x) = (\cdots((p_d x + p_{d-1})x + p_{d-2})\cdots)x + p_0$, which is a formula of size $2d + 1$. □

[9]This class may also include polynomials that actually depend on fewer variables but are masquerading as n-variate polynomials.

[10]Though this method was discovered at least 800 years earlier by the Iranian mathematician and astronomer Sharaf al-Dīn Ṭūsī (see Hogendijk [16]).

The following observation shows that the classes of algebraic formulas/ABPs/circuits are "robust" under some very natural operations; we shall call such models as *natural models*. These properties are precisely the ones that we rely on in this paper. Any *natural model* would be sufficient for our purposes but it might be convenient for the reader to focus on just the standard models of formulas, ABPs and circuits.

Definition 2.4 (Natural algebraic models)**.** An algebraic model $\mathcal{A}$ is called a *natural model* if the class of polynomials computed by it satisfies the following properties.

- Any polynomial of degree d with at most s monomials has $\mathcal{A}$-size at most $s \cdot d$. In the specific setting when the polynomial is a univariate, its $\mathcal{A}$-size is $O(d)$.
- Partial substitution of variables by constants does not increase the $\mathcal{A}$-size of any polynomial.
- If each of $Q_1, \ldots, Q_k$ is computable in $\mathcal{A}$-size s, then $\sum Q_i$ is computable in $\mathcal{A}$-size at most sk.
- Suppose $P(x_1, \ldots, x_n)$ is computable in $\mathcal{A}$-size s_1 and say $Q_1, \ldots, Q_n$ are polynomials each of which can be computed in $\mathcal{A}$-size s_2. Then, $P(Q_1, \ldots, Q_n)$ can be computed in $\mathcal{A}$-size at most $s_1 \cdot s_2$.

Observation 2.5. *Algebraic circuits, branching programs (ABPs), and formulas are natural algebraic models.*

2.2 Combinatorial designs

Definition 2.6 (Combinatorial designs (Nisan [23]))**.** A family of sets $\{S_1, \ldots, S_m\}$ is said to be an (ℓ, k, r) design if

- $S_i \subseteq [\ell]$,
- $|S_i| = k$,
- $\left|S_i \cap S_j\right| < r$ for any $i \neq j$.

The following is a standard construction of such designs based on the *Reed-Solomon* code, similar to the construction described in [24, Lemma 2.5].

Lemma 2.7 (Construction of designs)**.** *Let $c \geq 2$ be any positive integer. There is an algorithm that, given parameters ℓ, k, r satisfying $\ell = k^c$ and $r \leq k$ with k being a power of* 2*, outputs an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ for $m \leq k^{(c-1)r}$ in time* $\mathrm{poly}(m)$*.*

Proof. Since k is a power of 2, we can identify $[k]$ with the field $\mathbb{F}_k$ of k-elements and $[\ell]$ with $\mathbb{F}_k \times \mathbb{F}_{k^{c-1}}$. For each univariate polynomial $p(x) \in \mathbb{F}_{k^{c-1}}[x]$ of degree less than r, define the set S_p as

$$S_p = \{(i, p(i)) \;:\; i \in \mathbb{F}_k\}.$$

Since there are $k^{(c-1)r}$ such polynomials we get $k^{(c-1)r}$ subsets of $\mathbb{F}_k \times \mathbb{F}_{k^{c-1}}$ of size k each. Furthermore, since any two distinct univariate polynomials cannot agree at r or more places, it follows that $\left|S_p \cap S_q\right| < r$ for $p \neq q$. ◻

2.3 Hardness-randomness connections

Observation 2.8. *Let H be a hitting set for the class $\mathcal{C}(n, d, s)$ of n-variate polynomials of degree at most d that are computable by formulas of size s. Then, for any nonzero polynomial $Q(x_1, \ldots, x_n)$ such that $\deg(Q) \leq d$ and $Q(\mathbf{a}) = 0$ for all $\mathbf{a} \in H$, we have that Q cannot be computed by formulas of size s.*

Proof. If Q was indeed computable by formulas of size at most s, then Q is a member of $\mathcal{C}(n, d, s)$ for which H is a hitting set. This would violate the assumption that H was a hitting set for this class as Q is a nonzero polynomial in the class that vanishes on all of H. □

From this observation, it is easy to see that explicit hitting sets can be used to construct lower bounds.

Lemma 2.9 (Hitting sets to hardness [15, 1])**.** *Let H be an explicit hitting set for $\mathcal{C}(n, d, s)$. Then, for any $k \leq n$ such that $k|H|^{\frac{1}{k}} \leq d$, there is a polynomial $Q(z_1, \ldots, z_k)$ of individual degree smaller than $|H|^{\frac{1}{k}}$ that is computable in time $\mathrm{poly}(|H|)$ that requires formulas of size s to compute it. Furthermore, given the set H, there is an algorithm to output a formula of size $|H| \cdot d$ for Q in time $\mathrm{poly}(|H|)$.*

Proof. This is achieved by finding a nonzero k-variate polynomial, for $k \leq n$, of individual degree $d' < |H|^{\frac{1}{k}}$, that vanishes on the hitting set H; this can be done by interpreting it as a homogeneous linear system with $(d' + 1)^k$ "variables" and at most $|H|$ "constraints". Such a Q_k can then be found via interpolation by solving a system of linear equations in time $\mathrm{poly}(|H|)$. The degree of Q_k is at most $k \cdot |H|^{\frac{1}{k}} \leq d$ from the hypothesis and the hardness of Q_k follows from Observation 2.8. □

Remark 2.10 (Bit complexity of Q_k)**.** Note that we can obtain a hard polynomial Q_k such that its coefficients have bit-lengths that are at most polynomially large in terms of the (total) bit-lengths of the points in the given hitting set. Moreover, one can also ensure that such a computation only handles numbers of polynomially larger bit-lengths[11], thereby making Lemma 2.9 constructive in an *explicit* sense.

It is also known that we can get non-trivial hitting sets from suitable hardness assumptions. For a fixed (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ and a polynomial $Q(z_1, \ldots, z_k) \in \mathbb{F}[\mathbf{z}]$ we shall use the notation $Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ to denote the vector of polynomials

$$Q[\![\ell, k, r]\!]_{\mathrm{NW}} := (Q(\mathbf{y}\mid_{S_1}), Q(\mathbf{y}\mid_{S_2}), \ldots, Q(\mathbf{y}\mid_{S_m})) \in (\mathbb{F}[y_1, \ldots, y_\ell])^m .$$

Kabanets and Impagliazzo [18] showed that, if $Q(\mathbf{z}_{[k]})$ is hard enough, then $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}})$ is nonzero if and only if $P(\mathbf{x}_{[m]})$ is nonzero. However, their proof crucially relies on a result of Kaltofen [19] (or even a non-algorithmic version due to Bürgisser [7]) about the complexity of factors of polynomials. Hence, this connection is not directly applicable while working with other subclasses of circuits such as algebraic formulas as we do not know if they are closed

[11] Edmonds has shown that Gaussian elimination runs in polynomial time in terms of the total bit-length of the input [10], see [27, Theorem 3.3]. The same effect is achieved by Bareiss's algorithm [4], a modification of Gaussian elimination.

under factorization. The following lemma can be used in such settings and this paper makes heavy use of this.

Lemma 2.11 (Hardness to randomness without factor complexity)**.** *Let $Q(z_1, \ldots, z_k)$ be an arbitrary polynomial of individual degree smaller than d. Suppose there is an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ and a nonzero polynomial $P(x_1, \ldots, x_m)$, of degree at most D, that is computable by a formula of size at most s such that $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$. Then there is a polynomial $\tilde{P}(z_1, \ldots, z_k)$, whose degree is at most $k \cdot d \cdot D$ that is divisible by Q and computable by formulas of size at most $s \cdot (r-1) \cdot d^r \cdot (D+1)$.*

Moreover, if $r = 2$, then this upper bound can be improved to $4 \cdot s \cdot d \cdot (D+1)$

If the polynomial $Q(z_1, \ldots, z_k)$ in the above lemma was chosen such that Q vanished on some hitting set H for the class of size-s', n-variate, degree d' polynomials where $s' \geq s \cdot (r-1) \cdot d^r \cdot (D+1)$, then so does $\tilde{P}$ since Q divides it. If it happens that $\deg(\tilde{P}) \leq d'$, then Observation 2.8 immediately yields that $\tilde{P}$ cannot be computed by formulas of size s', contradicting the conclusion of the above lemma. Hence, in such instances, we would have that $P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \not\equiv 0$, without appealing to any result about closure of the particular model under factorization.

Proof. Borrowing the ideas from Kabanets and Impagliazzo [18], we look at the m-variate substitution $(x_1, \ldots, x_m) \mapsto Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ as a sequence of m univariate substitutions. We now introduce some notation to facilitate this analysis.

Given the (ℓ, k, r) design $\{S_1, \ldots, S_m\}$, let $\mathbf{y}_i = \mathbf{y}\mid_{S_i}$, for each $i \in [m]$. The tuple $Q[\![\ell, k, r]\!]_{\mathrm{NW}}$ can therefore be written as $(Q(\mathbf{y}_1), Q(\mathbf{y}_2), \ldots, Q(\mathbf{y}_m)) \in (\mathbb{F}[y_1, \ldots, y_\ell])^m$. For each $0 \leq i \leq m$, let

$$P_i = P(Q(\mathbf{y}_1), Q(\mathbf{y}_2), \ldots, Q(\mathbf{y}_i), x_{i+1}, \ldots, x_m)\,,$$

which is P after substituting for the variables $x_1, \ldots, x_i$. Since $P_0 = P$ is a nonzero polynomial and $P_m = P(Q[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, let t be the unique integer with $1 \leq t \leq m$, for which $P_{t-1} \not\equiv 0$ and $P_t \equiv 0$.

Since $P_t(\mathbf{y}, x_t, \ldots, x_m)$ is a nonzero polynomial, there exist values that can be substituted to the variables besides x_t and $\mathbf{y}_t$ such that it remains nonzero; let this polynomial be $P'_t(\mathbf{y}_t, x_t)$. Also, for each $j \in [t-1]$, let $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ be the polynomial obtained from $Q(\mathbf{y}_j)$ after this substitution, which is a polynomial of individual degree less than d on at most $(r-1)$ variables. We can now make the following observations about $P'(\mathbf{y}_t, x_t)$:

- Each $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ has a formula of size at most $(d(r-1)) \cdot d^{r-1}$, and thus $P'(\mathbf{y}_t, x_t)$ has a formula of size at most $(s \cdot (r-1) \cdot d^r)$,
- $\deg(P') \leq D \cdot \deg(Q) \leq D \cdot (kd)$, and $\deg_{x_t}(P') \leq D$,
- $P'(\mathbf{y}_t, Q(\mathbf{y}_t)) \equiv 0$.

The last observation implies that the polynomial $(x_t - Q(\mathbf{y}_t))$ divides P'. Therefore we can write $P' = (x_t - Q(\mathbf{y}_t)) \cdot R$, for some polynomial R. Consider P' and R as univariates in x_t with coefficients as polynomials in $\mathbf{y}_t$:

$$P' = \sum_{i=0}^{D} P'_i \cdot x_t^i \quad , \quad R = \sum_{i=0}^{D-1} R_i \cdot x_t^i.$$

If a is the smallest index such that $P'_a \neq 0$, then $P'_a = -R_a \cdot Q(\mathbf{y}_t)$ and hence $Q(\mathbf{y}_t)$ divides P'_a. Any coefficient P'_i can be obtained from P' using interpolation from $(D+1)$ evaluations of x_t. Hence, $\tilde{P} = P'_a$ can be computed in size $(s \cdot (r-1) \cdot d^r \cdot (D+1))$.

For the case of $r = 2$, observe that the polynomial $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ is a univariate of degree at most d. Thus, by Proposition 2.3, $Q^{(t)}(\mathbf{y}_j \cap \mathbf{y}_t)$ can be computed by a formula of size $2d + 1 \leq 4d$. So, we get an upper bound of $(4 \cdot s \cdot d)$ on the formula complexity of $P'(\mathbf{y}_t, x_t)$ (instead of $O(sd^2)$ that we would get by invoking the general bound for $r = 2$) and after interpolation as above, we get a bound of $4 \cdot s \cdot d \cdot (D+1)$ on the formula complexity of P'_a as defined above. ◻

3 Bootstrapping Hitting Sets

The following are the main bootstrapping lemmas to yield our main result. These lemmas follow the same template as in the proof of Agrawal, Ghosh and Saxena [2] but with some simple but important new ideas that avoid any requirement on bounds on factor complexity, and also permitting a result starting from a barely non-trivial hitting set.

Lemma 3.1 (Barely non-trivial to moderately non-trivial hitting sets). *Let $\epsilon > 0$ and $n \geq 2$ be constants. Suppose that for all large enough s there is an explicit hitting set of size $s^{n-\epsilon}$, for all degree-s, size-s algebraic formulas over n variables.*

Then for a large enough $m \geq n^8$, and for all $s \geq m$, there is an explicit hitting set of size $s^{\frac{m}{50}}$ for all degree-s, size-s algebraic formulas over m variables.

Lemma 3.2 (Bootstrapping moderately non-trivial hitting sets). *Let n_0 be large enough, and n be any power of two that is larger than n_0. Suppose for all $s \geq n$ there are explicit hitting sets of size $s^{g(n)}$ for $\mathcal{C}(n, s, s)$, the class of n-variate degree-s polynomials computed by size-s formulas.*

1. *Suppose $g(n) \leq \frac{n}{50}$, then for $m = n^{10}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) \leq \left(\frac{1}{2}\right) \cdot m^{\frac{1}{4}}$.*
2. *Suppose $g(n) \leq \left(\frac{1}{2}\right) \cdot n^{\frac{1}{4}}$, then for $m = 2^{n^{\frac{1}{4}}}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) = 20 \cdot \left(g(\log^4 m)\right)^2$.*

   Furthermore, $h(m)$ also satisfies $h(m) \leq \left(\frac{1}{2}\right) \cdot m^{\frac{1}{4}}$.

We will defer the proofs of these lemmas to the end of this section and complete the proof of Theorem 1.2.

Theorem 1.2 (Bootstrapping PIT for algebraic formulas, branching programs and circuits). *Let $\epsilon > 0$ and $n \geq 2$ be constants. Suppose that, for all large enough s, there is an explicit hitting set of size $s^{n-\epsilon}$ for all degree-s, size-s algebraic formulas (or algebraic branching programs or circuits) over n variables. Then, for all large s, there is an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for the class of degree-s, size-s algebraic formulas (or algebraic branching programs or circuits, respectively) over s variables.*

Proof. Notice that Lemma 3.1 and Lemma 3.2 are structured so that the conclusion of Lemma 3.1 is precisely the hypothesis of Lemma 3.2(1), the conclusion of Lemma 3.2(1) is precisely the hypothesis of Lemma 3.2(2), and Lemma 3.2(2) admits repeated applications as its conclusion also matches the requirements in the hypothesis. Thus, we can use one application of Lemma 3.1 followed by one application of Lemma 3.2(1) and repeated applications of Lemma 3.2(2) to get hitting sets for polynomials depending on larger sets of variables, until we can get a hitting set for the class $\mathcal{C}(s, s, s)$.

Let n_0 be large enough so as to satisfy the hypothesis of Lemma 3.1, and the two parts of Lemma 3.2. We start with an explicit hitting set of size $s^{n_0-\epsilon}$ for $\mathcal{C}(n_0, s, s)$ and one application of Lemma 3.1 gives an explicit hitting set of size $s^{\frac{n_1}{50}}$ for $\mathcal{C}(n_1, s, s)$ for $n_1 \geq n_0^8$ and all $s \geq n_1$. Using Lemma 3.2(1) we obtain an explicit hitting set of size $s^{(1/10)\cdot m_0^{\frac{1}{4}}}$ for the class $\mathcal{C}(m_0, s, s)$ for all $s \geq m_0 = n_1^{10}$. We are now in a position to apply Lemma 3.2(2) repeatedly. We now set up some basic notation to facilitate this analysis.

Suppose after i applications of Lemma 3.2(2) we have an explicit hitting set for the class $\mathcal{C}(m_i, s, s)$ of size s^{t_i}. We wish to track the evolution of m_i and t_i. Recall that $m_i = 2^{m_{i-1}^{\frac{1}{4}}}$ after one application of Lemma 3.2(2).

Let $\{b_i\}_i$ be such that $b_0 = \log m_0$ and, for every $i > 0$, let $b_i = 2^{(b_{i-1}/4)}$ so that $b_i = \log m_i$. Similarly to keep track of the complexity of the hitting set, if s^{t_i} is the size of the hitting set for $\mathcal{C}(m_i, s, s)$, then by Lemma 3.2(2) we have $t_0 = \left(\frac{1}{2}\right) m_0^{\frac{1}{4}}$ and $t_i = 20 \cdot t_{i-1}^2$ for all $i > 0$.
The following facts are easy to verify.

- $m_i \geq s$ or $b_i \geq \log s$ for $i = O(\log^* s)$,
- for all j, we have $t_j = 20^{(2^j-1)} \cdot t_0^{2^j} = \exp(\exp(O(j)))$.
- the exponent of s in the complexity of the final hitting set is $t_{O(\log^* s)} = \exp(\exp(O(\log^* s)))$.

Therefore we have an explicit hitting set of size $s^{\exp(\exp(O(\log^* s)))}$ for $\mathcal{C}(s, s, s)$. An explicit algorithm describing the hitting set generator is presented in Subsection 4.1. $\square$

3.1 Proofs of the bootstrapping lemmas

Here we prove the two main lemmas used in the proof of Theorem 1.2. We restate the lemmas here for convenience. The proofs follow a very similar template but with different settings of parameters and minor adjustments.

Lemma 3.3 (Barely non-trivial to moderately non-trivial hitting sets)**.** *Let $\epsilon > 0$ and $n \geq 2$ be constants. Suppose that for all large enough s there is an explicit hitting set of size $s^{n-\epsilon}$, for all degree-s, size-s algebraic formulas over n variables.*

Then for a large enough $m \geq n^8$, and for all $s \geq m$, there is an explicit hitting set of size $s^{\frac{m}{50}}$ for all degree-s, size-s algebraic formulas over m variables.

Proof. Let $a = \max(n, \frac{250}{\epsilon})$. We begin by fixing the design parameters, $k = n$, $\ell = a \cdot k^4 = a \cdot n^4$ and $r = 2$.

Constructing a suitably hard polynomial: For $B = \frac{5k}{\epsilon}$, we construct a polynomial $Q_k(z_1, \ldots, z_k)$ that vanishes on the hitting set for all size-s^B degree-s^B formulas over k variables, that has size $s^{B(k-\epsilon)}$ using Lemma 2.9. The polynomial $Q_k(\mathbf{z})$ has the following properties.

- Q_k has individual degree $d < s^{B(k-\epsilon)/k}$, and total degree $< k \cdot s^{B(k-\epsilon)/k}$.
- Q_k is not computable by formulas of size s^B.
- Q_k has a formula of size $\leq (kd) \cdot s^{B(k-\epsilon)}$.

Building the NW design: Using Lemma 2.7, we now construct an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m := \left(\frac{\ell}{k}\right)^r = \left(ak^{(4-1)}\right)^2 = a^2k^6$.

Variable reduction: Let $P(x_1, \ldots, x_m)$ be a nonzero m-variate degree-s polynomial computable by a formula of size s, and let $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$. Then, from the 'moreover' part of Lemma 2.11 (since $r = 2$), we get that there is a polynomial $\tilde{P}(z_1, \ldots, z_k)$ that vanishes on a hitting set for formulas of size s^B and degree s^B, and is computable by a formula of size at most

$$\begin{aligned}
\mathrm{size}(\tilde{P}) &\leq 4 \cdot s \cdot d \cdot (s+1) \\
&\leq 4s(s+1) \cdot s^{B(k-\epsilon)/k} \\
&\leq s^{\left(5 + \frac{B(k-\epsilon)}{k}\right)} = s^{5 + \frac{5k}{\epsilon} - 5} = s^B.
\end{aligned}$$

Moreover, note that the degree of $\tilde{P}(z_1, \ldots, z_k)$ is at most $(k \cdot d) \cdot s \leq s^{\left(2 + \frac{B(k-\epsilon)}{k}\right)} < s^B$. Since $\tilde{P}$ vanishes on the hitting set for formulas of size s^B and degree s^B, we get a contradiction due to Observation 2.8. Therefore, it must be the case that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ is nonzero.

Construction of the hitting set: Therefore, starting with a nonzero formula of degree s, size s, over m variables, we obtain a nonzero ℓ-variate polynomial of degree at most $s \cdot (kd) \leq s^B$. At this point we can just use the trivial hitting set given by the Polynomial Identity Lemma (Lemma 1.1) which has size at most $s^{B\ell}$.

Therefore, what remains to show is that our choice of parameters ensures that $B\ell < \frac{m}{50}$. This is true, as $\frac{m}{50} = \frac{a^2k^6}{50} = \frac{ak}{50} \cdot ak^5 = \left(\frac{1}{2}\right) \cdot \ell \cdot k > B\ell$.

The construction runs in time that is polynomial in the size of the hitting set in the conclusion, and the *bit-size* of the points in it. See Subsection 4.1 for a more elaborate discussion. □

Lemma 3.4 (Bootstrapping moderately non-trivial hitting sets)**.** *Let n_0 be large enough, and n be any power of two that is larger than n_0. Suppose for all $s \geq n$ there are explicit hitting sets of size $s^{g(n)}$ for $\mathcal{C}(n, s, s)$, the class of n-variate degree-s polynomials computed by size-s formulas.*

1. *Suppose $g(n) \leq \frac{n}{50}$, then for $m = n^{10}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) \leq \left(\frac{1}{2}\right) \cdot m^{\frac{1}{4}}$.*

2. *Suppose $g(n) \leq \left(\frac{1}{2}\right) \cdot n^{\frac{1}{4}}$, then for $m = 2^{n^{\frac{1}{4}}}$ and all $s \geq m$, there are explicit hitting sets of size $s^{h(m)}$ for $\mathcal{C}(m, s, s)$ where $h(m) = 20 \cdot \left(g(\log^4 m)\right)^2$.*

   Furthermore, $h(m)$ also satisfies $h(m) \leq \left(\frac{1}{2}\right) \cdot m^{\frac{1}{4}}$.

Proof. The proofs of both parts follow the same template as in the proof of Lemma 3.1 but with different parameter settings. Hence, we will defer the choices of the parameters ℓ, k, r towards the end to avoid further repeating the proof. For now, let ℓ, k, r be parameters that satisfy $r \leq k$, $\ell = k^2$ and $5r \cdot g(n) \leq k$.

Constructing a hard polynomial: The first step is to construct a polynomial $Q_k(z_1, \ldots, z_k)$ that vanishes on the hitting set for the class $\mathcal{C}(n, s^5, s^5)$, where[12] $k \leq n$. This can be done by using Lemma 2.9. The polynomial $Q_k(\mathbf{z})$ will therefore have the following properties.

- Q_k has individual degree d smaller than $s^{5g(n)/k}$, and degree at most $k \cdot s^{5g(n)/k}$.
- Computing Q_k requires formulas of size more than s^5.
- Q_k has a formula of size at most $s^{10g(n)}$.

Building the NW design: Using the parameters ℓ, k, r, and the construction from Lemma 2.7, we now construct an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m \leq k^r$.

Variable reduction using Q_k: Let $P(x_1, \ldots, x_m) \in \mathcal{C}(m, s, s)$ be a nonzero polynomial. Suppose for contradiction that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, then Lemma 2.11 states that there is a nonzero polynomial $\tilde{P}(z_1, \ldots, z_k)$ of degree at most $s \cdot k \cdot d$ such that Q_k divides $\tilde{P}$, and that $\tilde{P}$ can be computed by a formula of size at most

$$\begin{aligned} s \cdot (r-1) \cdot d^r \cdot (s+1) &\leq s^4 \cdot d^r \\ &\leq s^4 \cdot s^{5r \cdot g(n)/k} \\ &\leq s^5. && \text{(since } k, r \text{ satisfy } 5r \cdot g(n) \leq k) \end{aligned}$$

Furthermore, the degree of $\tilde{P}$ is at most $s \cdot r \cdot s^{5g(n)/k} \leq s^5$. Hence, $\tilde{P}$ is a polynomial in $k \leq n$ variables, of degree at most s^5 that vanishes on the hitting set of $\mathcal{C}(n, s^5, s^5)$ since Q_k divides $\tilde{P}$. But then, Observation 2.8 states that $\tilde{P}$ must require formulas of size more than s^5, contradicting the above size bound. Hence, it must be the case that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \not\equiv 0$.

Hitting set for $\mathcal{C}(m, s, s)$: At this point, we set the parameters k and r depending on how quickly $g(n)$ grows.

Part (1) $\left(g(n) \leq \frac{n}{50}\right)$: In this case, we choose $k = n$ and $r = 10$ (so we satisfy $5r \cdot g(n) \leq n = k$). From Lemma 2.7, we have an explicit (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m = k^r = n^{10}$.

[12] that is, Q_k is a k-variate polynomial that is just masquerading as an n-variate polynomial that does not depend on the last $n - k$ variables.

For any nonzero $P \in \mathcal{C}(m, s, s)$, we have that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ is a nonzero ℓ-variate polynomial of degree at most $s \cdot k \cdot s^{5g(n)/k} \leq s^3$. Hence, by just using the trivial hitting set via the Polynomial Identity Lemma (Lemma 1.1) we have an explicit hitting set of size $s^{3\ell} \leq s^{3m^{\frac{1}{5}}}$. Since $m \geq n_0$ and n_0 is large enough, we have that

$$h(m) := 3m^{\frac{1}{5}} \leq \left(\frac{1}{2}\right) \cdot m^{\frac{1}{4}}.$$

Part (2) $\left(g(n) \leq \left(\frac{1}{2}\right) n^{\frac{1}{4}}\right)$**:** In this case, we choose $k = \sqrt{n}$ and $r = n^{\frac{1}{4}}$, so that $5r \cdot g(n) \leq 10r \cdot g(n) \leq k$ and $\ell = n$. Using Lemma 2.7, we now construct an explicit (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m = 2^{n^{\frac{1}{4}}} \leq k^r$.

We have a formula computing the n-variate polynomial $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$ of size at most $s \cdot s^{10g(n)} \leq s^{20g(n)} =: s'$. Using the hypothesis for hitting sets for $\mathcal{C}(n, s', s')$, we have an explicit hitting set for $\mathcal{C}(m, s, s)$ of size at most

$$(s')^{g(n)} = s^{20g(n)^2} = s^{h(m)},$$

where $h(m) = 20\left(g((\log m)^4)\right)^2$. Since n_0 is large enough, we have that

$$\begin{aligned}
10 \cdot h(m) &\leq 20 \cdot 10 \cdot \left(g((\log m)^4)\right)^2 \\
&\leq 2\left(\log m\right)^2 && \left(\text{since } g(n) \leq \left(\frac{1}{2}\right) n^{\frac{1}{4}}\right) \\
&\leq m^{\frac{1}{4}}. && (\text{since } m \geq n_0 \text{ and } n_0 \text{ is large enough})
\end{aligned}$$

This completes the proof of both parts of the lemma. □

4 Finer analysis of the main result

This section addresses some of the subtle aspects about Theorem 1.2 as follows. First, we provide an algorithm outlining all the steps in obtaining the final hitting set from the initial one, which also helps us illustrate the explicitness of all the intermediate hitting sets (including the corresponding bit complexities). We then discuss how our theorem statement changes if the hypothesis holds only when the number of variables is a growing function of the size/degree s.

4.1 Algorithm for generating the hitting set

We now give an algorithm to generate an explicit hitting set for $\mathcal{C}(s, s, s)$, for all large s, using the hypothesis of Theorem 1.2. Let n_0 be the initial threshold from the hypothesis and let n_1 be a constant that satisfies the "large enough" requirements of Lemma 3.2.

$$
\begin{aligned}
& n_0 \geq 2 && t_0 = (n_0 - \epsilon) \\
& n_1 \text{ is large enough} && t_1 = \frac{n_1}{50} \\
& n_2 = n_1^{10} && t_2 = \left(\frac{1}{10}\right) n_2^{\frac{1}{4}} \\
\text{For all } i \geq 3, \quad & n_i = 2^{n_{i-1}^{\frac{1}{4}}} && t_i := 20 t_{i-1}^2
\end{aligned}
$$

We are provided an algorithm INITIAL-HITTING-SET(s) that outputs a hitting set for $C(n_0, s, s)$ of size at most $s^{n_0-\epsilon}$. Algorithm 1 describes a function HITTING-SET which, given inputs i and s, outputs a hitting set for $C(n_i, s, s)$ of size at most s^{t_i} in time poly(s^{t_i}).

From the growth of n_i, it follows that $n_b \geq s$ for $b = O(\log^* s)$ and $t_i = 20^{2^i-1} t_0^{2^i}$. Unfolding the recursion for HITTING-SET(j, s), for any j, the algorithm makes at most 2^j calls to INITIAL-HITTING-SET(s') for various sizes s' satisfying

$$
s' \leq s^{B \cdot 20^{j-1} \prod_{i=1}^{j-1} t_i} \leq s^{B t_{j-1}^2} = s^{O(t_j)}.
$$

Thus for HITTING-SET(b, s), the algorithm makes at most 2^b calls to INITIAL-HITTING-SET(s'), for sizes s' that are at most $s^{\exp(\exp(O(\log^* s)))}$. The overall running time is polynomial time in the size of the final hitting set which is $s^{t_b} = s^{\exp(\exp(O(\log^* s)))}$.

Bit complexity of the hitting sets. We will now discuss the bit complexity of the hitting sets that are generated during the bootstrapping procedure. We will analyze Algorithm 1 and show that any hitting set H for n-variate formulas that the algorithm outputs, will have a bit complexity that is at most $|H|^{f(n)}$, for $f(n) = \exp(O(\log^* n))$.

Let us first consider the case when $i \geq 3$. Suppose each evaluation point in $S_0 := H_{i-1}(s^5)$ is at most h^a bits long, where $h = |S_0|$ and $a = f(n_{i-1})$. Using Remark 2.10, we get that each coefficient of Q is at most h^{ca} bits long, for some other constant c. The output of Algorithm 1 for this case will be evaluations of Q on $S_1 := H_{i-1}(s^{20t_{i-1}})$. Since $|S_1| = h^{4t_{i-1}}$, the bit complexity of S_1 is at most $h^{4at_{i-1}}$. Now Q is a degree $< s^5$ polynomial with $O(h)$ monomials. As a result any evaluation of Q on a point from S_1 will have at most $O(\log h) \cdot \left(h^{ca} + s^5 \cdot h^{4at_{i-1}}\right) \leq h^{2a\cdot(4t_{i-1})} = |S_1|^{2a}$, as t_{i-1} is large enough. Since $n_i = 2^{n_{i-1}^{1/4}}$, we have that $f(n) = \exp(O(\log^* n))$. For the cases when $i = 1$ or $i = 2$, since we use the trivial hitting sets in place of S_1 in the above discussion, the same bounds will continue to hold.

We want to remark that although the bit complexity of the output of HITTING-SET(i, s) is not polynomial in the size s^{t_i}, the exponent t_i is always larger than $f(n_i)$. As a result the time taken to generate the final hitting set in the conclusion of Theorem 1.2 can still be bounded by $s^{\exp(\exp(O(\log^* s)))}$, albeit with a slightly larger constant in $O(\log^* s)$.

Algorithm 1: HITTING-SET

Input : Parameter i and a size s.
Output: A hitting set of size s^{t_i} size for $C(n_i, s, s)$.

if $i = 1$ **then**
- Let $A = \max\left(n_0, \frac{250}{\epsilon}\right)$, $B = \frac{3n_0}{\epsilon}$
- Let $H_0(s^B) :=$ INITIAL-HITTING-SET(s^B) `// size at most` $s^{B(n_0-\epsilon)}$
- Compute a nonzero polynomial Q in $k = n_0$ variables of individual degree smaller than $s^{Bt_0/k}$ that vanishes on $H_0(s^B)$. `// takes` $\mathrm{poly}(s^{Bt_0})$ `time`
- Compute an $(A \cdot n_0^4, n_0, 2)$-design $\{S_1, \ldots, S_{n_1}\}$.
- Let $S \subseteq \mathbb{F}$ be of size at least s^B.
- **return** $\left\{(Q[\![\ell,k,r]\!]_{\mathrm{NW}})(\mathbf{a}) \;:\; \mathbf{a} \in S^{An_0^5}\right\}$ `// size at most` $s^{BAn_0^5} \le s^{\frac{n_1}{50}} = s^{t_1}$

else if $i = 2$ **then**
- $H_1(s^5) :=$ HITTING-SET$(1, s^5)$ `// size at most` s^{5t_1}
- Compute a nonzero polynomial Q in $k = n_1$ variables of individual degree smaller than $s^{5t_1/k}$ that vanishes on $H_1(s^5)$. `// takes` $\mathrm{poly}(s^{5t_1})$ `time`
- Compute an $(n_1^2, n_1, 10)$-design $\{S_1, \ldots, S_{n_2}\}$.
- Let $S \subseteq \mathbb{F}$ be of size at least s^3.
- **return** $\left\{(Q[\![\ell,k,r]\!]_{\mathrm{NW}})(\mathbf{a}) \;:\; \mathbf{a} \in S^{n_1^2}\right\}$ `// size at most` $s^{3n_1^2} \le s^{0.1 \cdot n_2^{\frac{1}{4}}} = s^{t_2}$

else if $i \ge 3$ **then**
- $H_{i-1}(s^5) :=$ HITTING-SET$(i-1, s^5)$ `// size at most` $s^{5t_{i-1}}$
- Compute a nonzero polynomial Q in $k = \sqrt{n_{i-1}}$ variables of individual degree smaller than $s^{5t_{i-1}/k}$ that vanishes on $H_{i-1}(s^5)$. `// takes` $\mathrm{poly}(s^{5t_{i-1}})$ `time`
- Compute an $(n_{i-1}, \sqrt{n_{i-1}}, \sqrt[4]{n_{i-1}})$-design $\{S_1, \ldots, S_{n_i}\}$
- $H_{i-1}(s^{20t_{i-1}}) :=$ HITTING-SET$(i-1, s^{20t_{i-1}})$ `// size at most` $s^{20t_{i-1}^2}$
- **return** $\left\{(Q[\![\ell,k,r]\!]_{\mathrm{NW}})(\mathbf{a}) \;:\; \mathbf{a} \in H_{i-1}(s^{20t_{i-1}})\right\}$ `// size at most` $s^{20t_{i-1}^2} = s^{t_i}$

4.2 Bootstrapping with a slightly weaker hypothesis

We now discuss the analogue of Theorem 1.2 when the number of variables n in the hypothesis grows with the size parameter s. In particular, the hypothesis now guarantees a hitting set that is better than the brute force hitting set of size $(s+1)^n$ only when n grows faster than a certain function of s. For example, suppose that for all large n and s, the class $\mathcal{C}(n,s,s)$ has hitting sets of size[13] at most $s^{n^{\delta}(\log^{\circ 3} s)^{\delta}}$ for some small constant δ. Note that this hypothesis is not "helpful" when n is a constant, or even $(\log^{\circ 3} s)^{o(1)}$. Nevertheless, we shall show that even this scenario allows for a very similar bootstrapping procedure, and leads to hitting sets of size $s^{\mathrm{poly}(\log^{\circ 2} s)}$ for the class $\mathcal{C}(s,s,s)$.

We now state the analogue of the "single-step lemma" (Lemma 3.2) that helps us in proving our main theorem of this section (Theorem 4.2, which is proved at the end of the section).

Lemma 4.1. *Let $g,t : \mathbb{N} \to \mathbb{N}$ be non-decreasing functions such that $g(k) \le k^{1/4}$ and $t(x^{\log x}) \le 2t(x)$ for all x. Let $1 \le m \le s$ such that*

$$20 \cdot g\left((\log m)^4\right) \cdot t(s) \le \log m. \tag{4.1}$$

Let $(k_1, s_1) = \left((\log m)^4, s^5\right)$ and $(k_2, s_2) = \left((\log m)^4, s^{100 g((\log m)^2) t(s)}\right)$. Suppose $\mathcal{C}(k_1, s_1, s_1)$ and $\mathcal{C}(k_2, s_2, s_2)$ have explicit hitting sets of size at most $s_1^{g(k_1)t(s_1)}$ and $s_2^{g(k_2)t(s_2)}$, respectively.

Then, we have an explicit hitting set for $\mathcal{C}(m,s,s)$ of size at most $s^{g'(m)t'(s)}$, where

$$\begin{aligned} g'(m) &\le 400 \cdot \left(g\left((\log m)^4\right)\right)^2 \\ t'(s) &\le t(s)^2. \end{aligned}$$

Proof. The proof is exactly along similar lines, with just a little more care to set parameters appropriately.

Fix any m, s that satisfy equation (4.1). Set $r = \log m$, $k = (\log m)^2$ and $\ell = (\log m)^4$. With these parameters, equation (4.1) simplifies to $20 \cdot g(\ell) \cdot t(s) \le r$.

Let $S_1, \ldots, S_m$ be an (ℓ, k, r) design, as guaranteed by Lemma 2.7. Let $s_1 = s^5$. By the hypothesis for $(k_1 = \ell, s_1)$, there is an explicit hitting set H of size at most $s_1^{g(\ell)t(s_1)} \le s^{5g(\ell)\cdot 2\cdot t(s)} \le s^r$ for $\mathcal{C}(k, s_1, s_1)$. Let $Q(z_1, \ldots, z_k)$ be the explicit polynomial of individual degree at most $s^{10 g(\ell) t(s)/k} \le s^{r/k}$, as guaranteed by Lemma 2.9, that vanishes on the set H. Therefore, Q_k satisfies the following properties:

- Q_k has individual degree less than $s^{r/k}$ and total degree at most $k \cdot s^{r/k}$.
- Q_k can be computed by formulas of size at most $s^{20 g(\ell) t(s)} \le s^r$.

Now, suppose $P(x_1, \ldots, x_m)$ is any nonzero polynomial in $\mathcal{C}(m,s,s)$ but $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, then Lemma 2.11 states that there is a nonzero polynomial $\tilde{P}(z_1, \ldots, z_k)$ such that

[13] Here $\log^{\circ i}$ denotes i iterated applications of log, i. e. $\log^{\circ 3} s = \log\log\log s$.

- $\tilde{P}$ has degree at most s^3, and is a multiple of Q_k
- $\tilde{P}$ is computed by formulas of size at most $s^3 \cdot s^{r^2/k} = s^4$

However, Observation 2.8 asserts that $\tilde{P}$ must require formulas of size at least s^5 and hence it must be the case that $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \not\equiv 0$.

Let $P'(z_1, \ldots, z_\ell) = P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}})$. This is an ℓ-variate nonzero polynomial of degree at most $s \cdot \deg(Q_k) \le s^3$ and is computable by formulas of size $s_2 = s \cdot s^{20g(\ell)t(s)} \le s^{20g(\ell)t(s)}$. Since $20g(\ell)t(s) \le \log m \le \log s$, we have that $t(s_2) \le 2t(s)$. Using the hypothesis for $(k_2 = \ell, s_2)$ an explicit hitting set for $C(\ell, s_2, s_2)$ of size at most $s_2^{g(\ell)t(s_2)}$ which can be bounded above as follows:

$$\begin{aligned} s_2^{g(\ell)t(s_2)} &\le s_2^{2\cdot g(\ell)t(s)} \\ &\le s^{20\cdot g(\ell)t(s)\cdot 2\cdot g(\ell)\cdot t(s)} \\ &\le s^{400\cdot g(\ell)^2\cdot t(s)^2} \le s^{g'(m)t'(s)}. \end{aligned}$$

Therefore, we have an explicit hitting set for $C(m, s, s)$ of size at most $s^{g'(m)t'(s)}$. □

Intuitively, the above lemma states that if $g(k)t(s)$ is substantially smaller than k, then explicit hitting sets of size $s^{g(k)t(s)}$ for $C(k, s, s)$ can be used to obtain an $s^{g'(m)t'(s)}$ hitting set for $C(m, s, s)$ where m is exponentially larger than k (if s is not too large in comparison to m). Suppose $g'(m)t'(s)$ is also substantially smaller than m (which would roughly translate to $g(k)t(s)$ being *much* smaller than k; say something like $\log\log k$), then we may be in a position to use the lemma again to get an even better hitting set.

The following theorem sets up the parameters suitably when we are in a position to use the above lemma multiple times.

Theorem 4.2. *Let $g_0, t_0 : \mathbb{N} \to \mathbb{N}$ be non-decreasing functions with $g_0(m) \le m^{1/4}$ and $t_0(s) \le \log s$. Let $g_1, g_2, \ldots : \mathbb{N} \to \mathbb{N}$ and $t_1, t_2, \ldots : \mathbb{N} \to \mathbb{N}$ be defined as $g_{i+1}(m) = 400 \cdot \left(g_i\left((\log m)^4\right)\right)^2$ and $t_{i+1}(s) = (t(s))^2$. Furthermore, let $L_0 : \mathbb{N} \to \mathbb{N}$ as $L_1(x) = (\log x)^2$ and $L_{i+1}(x) = (L_i((\log x)^4))^2$ for all $i \ge 1$.*

Suppose for every $m, s \ge 1$ we have an explicit hitting set for $C(m, s, s)$ of size at most $s^{g_0(m)\cdot t_0(s)}$. Let $r_{m,s} \ge 1$ be the largest index such that

- *$g_r(x) \le x^{1/4}$ and $t_r(x^{\log x}) \le 2 \cdot t_r(x)$ for all $x \ge 1$,*
- *$100 \cdot g_r(m) \cdot t_r(s) \le L_r(m)$.*

Then for any $j \le r_{m,s}$, we have an explicit hitting set for $C(m, s, s)$ of size at most $s^{g_j(m)t_j(s)}$.

In particular, $C(s, s, s)$ has an explicit hitting set of size $s^{L_r(s)}$ for $r = r_{s,s}$.

Although the above theorem is stated in a general form, it would be instructive to just consider *nice* functions of the form $g_0(s) = s^{0.1}$. In this case, it can be seen that

$$\begin{aligned} g_i(s) &= 2^{2^{O(i)}} \cdot \mathrm{poly}(\log^{\circ i} s), \\ t_i(s) &= (\log^{\circ 3} s)^{2^i}, \text{ and} \\ L_i(s) &= 2^{2^{O(i)}} \cdot \mathrm{poly}(\log^{\circ i} s), \end{aligned}$$

where $g_1(s) \ll L_1(s)$ but, certainly for $i = \log^* s$ we have $g_i(s) > L_i(s)$. If t_0 is the constant function, then we are essentially in the same regime as in the previous sections, so we get $r_{s,s} = \Theta(\log^* s)$, and we recover Theorem 1.2. However, if suppose $t_0(s) = \log^{\circ 3} s$ (where $\log^{\circ i} s$ is the iterated logarithm function, i. e. $\log^{\circ 3} s = \log\log\log s$), then $t_3(s) = (\log^{\circ 3} s)^{2^3} > L_3(s)$ and hence $r_{s,s} = 2$. Nevertheless, the above theorem yields an explicit hitting set of size $C(s,s,s)$ of size at most $s^{\mathrm{poly}\log^{\circ 2} s}$ which is still substantially better than $s^{g_0(s)t_0(s)} > s^{s^{0.1}}$.

Proof. We will prove the theorem by induction on j. For any m, s, the base case of $j = 0$ trivially satisfied by the hypothesis of the theorem and hence there is nothing to prove.

Inductive step ($j \to j+1$): Suppose $g_{j+1}(x) \le x^{1/4}$ and $t_{j+1}(x^{\log x}) \le 2 \cdot t_{j+1}(x)$ for all $x \ge 1$. For each $i \le j+1$, define $\mathcal{D}_i = \{(m,s) : m \le s \ , \ 100 \cdot g_i(m) t_i(s) \le L_i(m)\}$. By the induction hypothesis, for every $(k, s') \in \mathcal{D}_j$ we have an explicit hitting set for $C(k, s', s')$ of size at most $s'^{g_j(k)t_j(s')}$.

Fix any $(m,s) \in \mathcal{D}_{j+1}$. To construct the explicit hitting set for $C(m,s,s)$, we will use Lemma 4.1. In order to do that, we need to assert the following claims:

(a) (m,s) satisfy equation (4.1) (for $g = g_j$ and $t = t_j$),

(b) $C(\ell, s_1, s_1)$, for $\ell = (\log m)^4$ and $s = s^5$, has an explicit hitting set of size at most $s_1^{g_j(\ell)t_j(s)}$, and

(c) $C(\ell, s_2, s_2)$, for $\ell = (\log m)^4$ and $s_2 = s^{20 g_j(\ell) t_j(s)}$, has an explicit hitting set of size at most $s_2^{g_j(\ell)t_j(s_2)}$.

Pf of Claim (a): $(m,s) \in \mathcal{D}_{j+1}$ implies that

$$\begin{aligned} 20 \cdot g_j((\log m)^4) \cdot t_j(s) &= 20 \cdot \left(\frac{g_{j+1}(m)}{400}\right)^{1/2} \cdot \left(t_{j+1}(m)\right)^{1/2} \\ &= \left(g_{j+1}(m) t_{j+1}(s)\right)^{1/2} \\ &\le \frac{1}{10} \cdot L_{j+1}(m)^{1/2} \le \log m. \end{aligned}$$

Pf of Claim (b): If $\ell = (\log m)^4$ and $s_1 = s^5$,

$$\begin{aligned} 100 \cdot g_j(\ell)t_j(s_1) &\le 200 \cdot g_j(\ell)t_j(s) \\ &= 10 \cdot \left(g_{j+1}(m) \cdot t_{j+1}(s)\right)^{1/2} \\ &\le \frac{10}{10} \cdot L_{j+1}(m)^{1/2} \\ &= L_j(\ell). \end{aligned}$$

Hence $(\ell, s_1) \in \mathcal{D}_j$ and by the inductive hypothesis, we have an explicit hitting set for $C(\ell, s_1, s_1)$ of size at most $s_1^{g_j(\ell)t_j(s_1)}$.

Pf of Claim (c): For $\ell = (\log m)^4$ and $s_2 = s^{20 g_j(\ell)t_j(s)}$, note that Claim (a) shows that $20 g_j(\ell)t_j(s) \le \log m \le \log s$ and hence $t_j(s_2) \le 2t_j(s)$. Therefore,

$$\begin{aligned} 100 \cdot g_j(\ell)t_j(s_2) &\le 200 \cdot g_j(\ell)t_j(s) \\ &= 10 \cdot \left(g_{j+1}(m) \cdot t_{j+1}(s)\right)^{1/2} \\ &= \left(L_{j+1}(m)\right)^{1/2} \le L_j(\ell). \end{aligned}$$

Hence $(\ell, s_2) \in \mathcal{D}_j$ and, by the induction hypothesis, we have an explicit hitting set for $C(\ell, s_2, s_2)$ of size at most $s_2^{g_j(\ell)t_j(s_2)}$.

Using Lemma 4.1, we have that $C(m, s, s)$ has an explicit hitting set of size $s^{g_{j+1}(m)t_{j+1}(s)}$. □

5 Open problems

We conclude with some open questions.

- A natural question in the spirit of the results in this paper, and those in [2] seems to be the following: Can we hope to bootstrap lower bounds? In particular, can we hope to start from a mildly non-trivial lower bound for general algebraic circuits (e. g., superlinear or just superpolynomial), and hope to amplify it to get a stronger lower bound (superpolynomial or truly exponential, respectively). In the context of non-commutative algebraic circuits, Carmosino, Impagliazzo, Lovett and Mihajlin [8] recently showed such results, but no such result appears to be known for commutative algebraic circuits.

- It would also be interesting to understand if it is possible to bootstrap white box PIT algorithms. The proofs in this paper strongly rely on the fact that we have a non-trivial blackbox derandomization of PIT.

- Kabanets and Impagliazzo [18] show that given a polynomial family which requires exponential-size circuits, there is a blackbox PIT algorithm which runs in quasipolynomial time ($\exp(\mathrm{poly}(\log n))$) on circuits of size $\mathrm{poly}(n)$ and degree $\mathrm{poly}(n)$, where n is the number of variables. Thus, even with the best hardness possible, the running time of the

PIT algorithm obtained is still no better than quasipolynomially bounded. The question is to improve this running time to get a better upper bound than that obtained in [18]. In particular, can we hope to get a deterministic polynomial-time PIT assuming that we have explicit polynomial families of exponential hardness. This seems to be closely related to the question about bootstrapping lower bounds.

Since the first version of this paper appeared, there has been some work related to the problems studied in this paper. Specifically, Guo, Kumar, Saptharishi and Solomon [14] showed that it is possible to bootstrap even explicit hitting sets of size $s^n - 1$ for n-variate circuits of size s and degree s, for a constant n, to obtain polynomial-time PIT for s-variate circuits of size and degree s. The proof in [14] goes via the direct construction of a hitting set generator with constant seed length assuming a sufficiently strong hardness assumption, thereby making progress on the last open question state above as well. One caveat of the results in [14] is that the underlying field needs to be of sufficiently large characteristic or characteristic zero. Proving analogous results over fields of small characteristic continues to be a very interesting open problem.

Acknowledgments: Ramprasad and Anamay would like to thank the organizers of the Workshop on Algebraic Complexity Theory (WACT 2018) where we first started addressing this problem. Ramprasad and Anamay would also like to thank Suhail Sherif and Srikanth Srinivasan for making an observation which led to the strengthening of an older version of this paper. Mrinal is thankful to Michael Forbes, Josh Grochow and Madhu Sudan for insightful discussions.

We also thank the anonymous reviewers at Theory of Computing for valuable feedback. The discussion in subsection 4.2 in particular was a result of a query from one of the reviewers on whether an analogue of Theorem 1.2 continues to hold when the parameter n in the hypothesis is growing with s.

Ramprasad and Anamay would also like to acknowledge the support from the Department of Atomic Energy, Government of India, under project number RTI4001.

References

[1] MANINDRA AGRAWAL: Proving lower bounds via pseudo-random generators. In *Proc. 25th Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'05)*, pp. 92–105. Springer, 2005. [doi:10.1007/11590156_6] 3, 6, 7, 12

[2] MANINDRA AGRAWAL, SUMANTA GHOSH, AND NITIN SAXENA: Bootstrapping variables in algebraic circuits. *Proc. Nat. Acad. Sci. USA*, 116(17):8107–8118, 2019. Preliminary version in STOC'18. [doi:10.1073/pnas.1901272116, ECCC:TR18-035] 4, 5, 8, 14, 24

[3] MANINDRA AGRAWAL, NEERAJ KAYAL, AND NITIN SAXENA: PRIMES is in P. *Ann. Math.*, 160(2):781–793, 2004. [doi:10.4007/annals.2004.160.781] 3

[4] ERWIN H. BAREISS: Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comput.*, 22(103):565–578, 1968. [doi:10.1090/S0025-5718-1968-0226829-0] 12

[5] WALTER BAUR AND VOLKER STRASSEN: The complexity of partial derivatives. *Theoret. Comput. Sci.*, 22(3):317–330, 1983. [doi:10.1016/0304-3975(83)90110-X] 2

[6] ANURAG BISHNOI, PETE L. CLARK, ADITYA POTUKUCHI, AND JOHN R. SCHMITT: On zeros of a polynomial in a finite grid. *Combin. Probab. Comput.*, 27(3):310–333, 2018. [doi:10.1017/S0963548317000566] 3

[7] PETER BÜRGISSER: *Completeness and Reduction in Algebraic Complexity Theory*. Volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000. [doi:10.1007/978-3-662-04179-6] 12

[8] MARCO L. CARMOSINO, RUSSELL IMPAGLIAZZO, SHACHAR LOVETT, AND IVAN MIHAJLIN: Hardness amplification for non-commutative arithmetic circuits. In *Proc. 33rd Comput. Complexity Conf. (CCC'18)*, pp. 12:1–16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.CCC.2018.12, ECCC:TR18-095] 5, 24

[9] RICHARD A. DEMILLO AND RICHARD J. LIPTON: A probabilistic remark on algebraic program testing. *Inform. Process. Lett.*, 7(4):193–195, 1978. [doi:10.1016/0020-0190(78)90067-4] 3

[10] JACK EDMONDS: Systems of distinct representatives and linear algebra. *J. Res. Nat. Bureau of Standards (B)*, 71:241–245, 1967. [doi:10.6028/jres.071b.033] 12

[11] STEPHEN A. FENNER, ROHIT GURJAR, AND THOMAS THIERAUF: Bipartite perfect matching is in quasi-NC. *SIAM J. Comput.*, 50(3), 2021. Preliminary version in STOC'16. [doi:10.1137/16M1097870, arXiv:1601.06319, ECCC:TR15-177] 3

[12] MICHAEL A. FORBES: *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. Ph. D. thesis, Massachusetts Institute of Technology, 2014. DSpace@MIT. 2

[13] JOSHUA GROCHOW: Degree restriction for polynomials in VP, 2013. stackexchange. 2

[14] ZEYU GUO, MRINAL KUMAR, RAMPRASAD SAPTHARISHI, AND NOAM SOLOMON: Derandomization from algebraic hardness. *SIAM J. Comput.*, 51(2):315–335, 2022. Preliminary version in FOCS'19. [doi:10.1137/20M1347395] 25

[15] JOOS HEINTZ AND CLAUS-PETER SCHNORR: Testing polynomials which are easy to compute (extended abstract). In *Proc. 12th STOC*, pp. 262–272. ACM Press, 1980. [doi:10.1145/800141.804674] 3, 6, 7, 12

[16] JAN P. HOGENDIJK: Sharaf al-dīn Ṭūsī on the number of positive roots of cubic equations. *Historia Mathematica*, 16(1):69–85, 1989. [doi:10.1016/0315-0860(89)90099-2] 10

[17] MAURICE J. JANSEN AND RAHUL SANTHANAM: Marginal hitting sets imply super-polynomial lower bounds for permanent. In *Proc. 3rd Innovations in Theoret. Comp. Sci. Conf. (ITCS'12)*, pp. 496–506. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012. [doi:10.1145/2090236.2090275, ECCC:TR11-133] 5, 9

[18] VALENTINE KABANETS AND RUSSELL IMPAGLIAZZO: Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complexity*, 13(1–2):1–46, 2004. Preliminary version in STOC'03. [doi:10.1007/s00037-004-0182-6, ECCC:TR02-055] 4, 6, 7, 8, 12, 13, 24, 25

[19] ERICH KALTOFEN: Factorization of polynomials given by straight-line programs. In SILVIO MICALI, editor, *Randomness and Computation*, pp. 375–412. JAI Press, 1989. DSpace@MIT. Preliminary version in FOCS'85. 7, 12

[20] MRINAL KUMAR, RAMPRASAD SAPTHARISHI, AND ANAMAY TENGSE: Near-optimal bootstrapping of hitting sets for algebraic circuits. In *Proc. 30th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'19)*, pp. 639–646. SIAM, 2019. [doi:10.1137/1.9781611975482.40] 1

[21] CARSTEN LUND, LANCE FORTNOW, HOWARD J. KARLOFF, AND NOAM NISAN: Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1990. Preliminary version in FOCS'90. [doi:10.1145/146585.146605] 3

[22] KETAN MULMULEY, UMESH V. VAZIRANI, AND VIJAY V. VAZIRANI: Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. Preliminary version in STOC'87. [doi:10.1007/BF02579206] 3

[23] NOAM NISAN: Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. [doi:10.1007/BF01375474] 11

[24] NOAM NISAN AND AVI WIGDERSON: Hardness vs randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. [doi:10.1016/S0022-0000(05)80043-1] 7, 11

[25] ØYSTEIN ORE: Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. 3

[26] RAMPRASAD SAPTHARISHI: A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015. 2

[27] ALEXANDER SCHRIJVER: *Theory of Linear and Integer Programming*. Wiley–Interscience, 1986. Wiley. 12

[28] JACOB T. SCHWARTZ: Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. [doi:10.1145/322217.322225] 3

[29] ADI SHAMIR: IP=PSPACE. *J. ACM*, 39(4):869–877, 1992. Preliminary version in FOCS'90. [doi:10.1145/146585.146609] 3

[30] AMIR SHPILKA AND AMIR YEHUDAYOFF: Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comp. Sci.*, 5(3–4):207–388, 2010. [doi:10.1561/0400000039] 2

[31] AMIT SINHABABU AND THOMAS THIERAUF: Factorization of polynomials given by arithmetic branching programs. *Comput. Complexity*, 30(2):15:1–47, 2021. Preliminary version in CCC'20. [doi:10.1007/s00037-021-00215-0, ECCC:TR20-077] 8

[32] OLA SVENSSON AND JAKUB TARNAWSKI: The matching problem in general graphs is in quasi-NC. In *Proc. 58th FOCS*, pp. 696–707. IEEE Comp. Soc., 2017. [doi:10.1109/FOCS.2017.70, arXiv:1704.01929, ECCC:TR17-059] 3

[33] RICHARD ZIPPEL: Probabilistic algorithms for sparse polynomials. In *Proc. Internat. Symp. Symbolic and Algebraic Manipulation (EUROSAM'79)*, pp. 216–226. Springer, 1979. [doi:10.1007/3-540-09519-5_73] 3

AUTHORS

Mrinal Kumar
Reader
School of Technology and Computer Science
Tata Institute of Fundamental Research, Mumbai
mrinal@tifr.res.in
`https://mrinalkr.bitbucket.io/`

Ramprasad Saptharishi
Associate Professor
School of Technology and Computer Science, TIFR, Mumbai
ramprasad@tifr.res.in
`https://www.tifr.res.in/~ramprasad/`

Anamay Tengse
Postdoctoral Student
Efi Arazi School of Computer Science, Reichman Universiry, Herzliya
anamay.tengse@gmail.com
`https://anamay.bitbucket.io/`

ABOUT THE AUTHORS

MRINAL KUMAR is a faculty member in the School of Technology and Computer Science at the Tata Institute of Fundamental Research (TIFR), Mumbai. Before moving to TIFR in 2022, he spent a few years as an assistant professor in the Department of Computer Science and Engineering at IIT Bombay. He is generally interested in questions in Computational Complexity, Algebraic Complexity and Coding Theory. He did his Ph. D. at Rutgers, where he was advised by Swastik Kopparty and Shubhangi Saraf, and spent his postdoctoral years at a convex combination of Harvard, the Simons Institute and the University of Toronto before moving to Mumbai in 2019. Much before that, he grew up in the then small town of Deoghar in the state of Jharkhand in India, and for the first fifteen years of his life dreamed of growing up to play cricket for a living. His interest in math was kindled during the course of his interaction with Dr. Jugal Kishore Singh, his math teacher in high school. Dr. Singh convinced him (and half his teammates in the cricket team!) that math could be interesting, challenging and yet accessible. His unconventional lectures were full of delightful anecdotes, fantastic questions and quite remarkably, even his exams were enjoyable, non-scary and yet interesting. Perhaps most importantly, he encouraged his students to think independently, which many of his former students believe is the most important thing that they ever learnt in school.

RAMPRASAD SAPTHARISHI is a faculty member in the School of Technology and Computer Science at the Tata Institute of Fundamental Research, Mumbai, and is generally interested in most questions of an algebraic nature. He did his Ph. D. at Chennai Mathematical Institute, advised by Prof. Manindra Agrawal, and spent a couple of years at Microsoft Research, India and Tel Aviv University for his postdoc. His other interests include board games, reading ridiculously long web-series and writing software code. However, writing reasonable biographies continues to remain outside his area of expertise.

Anamay Tengse is currently a postdoc at Reichman University in Herzliya, and was a postdoc at the University of Haifa for a couple of years before that. This work was done when he was a Ph. D. student in the School of Technology and Computer Science at TIFR, Mumbai, advised by Ramprasad Saptharishi. Prior to that, he obtained an M. Tech. from the Department of Computer Science and Engineering at IIT Bombay, where he first got interested in the theoretical aspects of computing. His interest in research comes largely from his interactions with the faculty and his friends from Goa College of Engineering, where he did his undergrad. He spends a dangerously large chunk of his time indulging in music, cricket, quizzes on sporcle, trying to understand poetry and linguistics, and justifying his lack of interest in football (soccer) to people who question his Goan origin.