Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 132 (2018)

Near-optimal Bootstrapping of Hitting Sets for Algebraic Models

Mrinal Kumar[*] Ramprasad Saptharishi[†] Anamay Tengse[‡]

September 5, 2018

Abstract

The classical lemma of Ore-DeMillo-Lipton-Schwartz-Zippel [Ore22, DL78, Zip79, Sch80] 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 a 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:

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

  That is, if we can obtain a barely non-trivial exponent compared to the trivial $(s+1)^n$ sized 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 [AGS18] 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 branching programs and formulas.

[*] mrinalkumar08@gmail.com. Simons Institute for the Theory of Computing, Berkeley, USA. A part of this work was done during a postdoctoral stay at Center for Mathematical Sciences and Applications at Harvard and while visiting TIFR, Mumbai.

[†] ramprasad@tifr.res.in. Tata Institute of Fundamental Research, Mumbai, India. Research supported by Ramanujan Fellowship of DST

[‡] tengse.anamay@tifr.res.in. Tata Institute of Fundamental Research, Mumbai, India. Supported by a fellowship of the DAE.

ISSN 1433-8092

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 [For14, Section 3.2] and Grochow's answer [Gro] 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 (cf. surveys [SY10, Sap15]), the current best lower bound for general algebraic circuits is merely an $\Omega(n \log d)$ lower bound of Baur and Strassen [BS83].

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 of its own right, and many classical results such as the primality testing algorithm [AKS04], $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN90, Sha90], algorithms for graph matching [MVV87, FGT16, ST17] all have a polynomial identity test at its core.

This algorithmic question has two flavours: 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

[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.

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 $\mathrm{poly}(|H|)$.

The classical Ore-DeMillo-Lipton-Schwartz-Zippel Lemma [Ore22, DL78, Zip79, Sch80] states that any nonzero polynomial $f(x_1, \ldots, x_n)$ of degree at most d will evaluate to a nonzero value at a randomly chosen point from a grid $S^n \subseteq \mathbb{F}^n$ with probability at least $1 - \frac{d}{|S|}$. Therefore, 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) $\mathrm{poly}(s)$ sized 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 [HS80], and Agrawal [Agr05] 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 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 $2^{\Omega(n)}$ sized circuits. This connection between PIT and lower bounds was strengthened further by Kabanets and Impagliazzo [KI04] 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 [AGS18] 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 $s^{\exp \circ \exp(O(\log^* s))}$. 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 [AGS18] 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 size s and n-variate 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 Ore-DeMillo-Lipton-Schwartz-Zippel lemma [Ore22, DL78, Zip79, Sch80]. 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 1.1 (Bootstrapping PIT for algebraic formulas, branching programs and circuits)**.** *Let $\varepsilon > 0$ be a constant. For some large enough n suppose that, for all $s \geq n$, there is an explicit hitting set of size $s^{n-\varepsilon}$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over n variables. Then, there is an explicit hitting set of size $s^{\exp\circ\exp(O(\log^* s))}$ for the class of degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over s variables.*

Note that $(s+1)^{n-\varepsilon} = s^{n-\varepsilon} \cdot \left(1+\frac{1}{s}\right)^{n-\varepsilon} < e \cdot s^{n-\varepsilon} < s^{n-\varepsilon'}$ for some other constant $\varepsilon' > 0$ since s is large enough. Hence, for this theorem, there is no qualitative difference if the hitting set had size $(s+1)^{n-\varepsilon}$ instead of $s^{n-\varepsilon}$. We also note that as far as we understand, such a statement for classes such as algebraic branching programs or formulas, even with the stronger hypothesis of there being a $s^{O(n^{(1/2)-\varepsilon})}$, did not follow from the results of Agrawal et al. [AGS18]. 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 1.2 (From slightly non-trivial PIT to lower bounds)**.** *Let $\varepsilon > 0$ be a constant. For some large enough n suppose that, for all $s \geq n$, there is an explicit hitting set of size $(s^{n-\varepsilon})$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over n variables. Then, for every function $d : \mathbb{N} \to \mathbb{N}$, there is a polynomial family $\{f_n\}$, where f_n is n variate and degree $d(n)$, and for every large enough n, f_n cannot be computed by algebraic formulas (algebraic branching programs or circuits respectively) of size smaller than $\binom{n+d}{d}^{1/\exp\circ\exp(O(\log^* nd))}$. 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. In a recent result, Carmosino et al. [CILM18] showed that given an explicit polynomial family of constant degree which requires super linear sized 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 [CILM18], 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 1.2 and the main result of [CILM18] 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 1.3), however the main result of [CILM18] 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.

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 et al. [AGS18], comes from the results of Kabanets and Impagliazzo [KI04], and those of Heintz and Schnorr [HS80] and Agrawal [Agr05]. We start by informally stating these results.

Theorem 1.3 (Informal, Heintz and Schnorr [HS80], Agrawal [Agr05]). *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 on n 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 on k variables, 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 [KI04] that shows that hardness can lead to derandomization.

Theorem 1.4 (Informal, Kabanets and Impagliazzo [KI04]). *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 $\mathrm{poly}(d)^{n^{\varepsilon}}$ for every $\varepsilon > 0$.*

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 1.1 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 1.3, which simply uses interpolation to find a nonzero polynomial Q on 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 Nisan-Wigderson design (see Definition 2.5) $\{S_1, S_2, \ldots, S_m\}$, 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 1.4, 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 branching programs and formulas. In particular, we do not need Kaltofen's seminal result that algebraic circuits are closed under polynomial factorization, whereas the proof of Kabanets et al. crucially uses Kaltofen's result [Kal89]. This come from the simple, yet crucial, 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.

- **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 $\ell = \text{poly}\log m$, 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 1.2, we let $t(n) = \exp \circ \exp(O(\log^* n))$. Now, we invoke the the conclusion of Theorem 1.1 with $s = \binom{n+d}{d}^{1/10t(n)}$. Thus, we get an explicit hitting set H of size $\binom{n+d}{d}^{1/10}$ for n variate circuits of size s and degree d. We now use Theorem 1.3 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.

Similarities and differences with the proof of Agrawal et al. [AGS18]. The high level outline of our proof is essentially the same as that of Agrawal et al. [AGS18]. However, there are some differences that make our final arguments shorter, simpler and more robust than those of Agrawal et al. thus leading to a stronger and near optimal boostrapping statement in Theorem 1.1. Moreover, as we already alluded to, our proof extends to formulas and algebraic branching programs as well, whereas, to the best of our understanding, those of Agrawal et al. [AGS18] do not. We now elaborate on the differences.

One of the main differences between the proofs in this paper and those of Agrawal et al. is in the use of the the result of Kabanets and Impagliazzo [KI04]. Agrawal et al. use this result as a blackbox to get deterministic PIT using hard polynomials. The result of Kabanets et al. [KI04] crucially relies on a result of Kaltofen, which shows that low degree algebraic circuits are closed under polynomial factorization i.e. 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 algebraic branching programs or formulas, and hence the results of Agrawal et al. in [AGS18] do not seem to extend to these settings. Also, the removal of any dependence on the "factorization exponent" e is crucial in our proof as it allows us to start with a hypothesis of a barely non-trivial hitting set. The other main difference between our proof and that of Agrawal et al. is rather technical but we try to briefly describe it. This is in the choice of Nisan-Wigderson 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 Agrawal et al.[3]

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 1.1 and also those of the results in the work of Agrawal et al. [AGS18] 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 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 1.1 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.

Remark. *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.* ◊

2 Preliminaries

Notation

- For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

[3] 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 $s^{n^{1-\delta}}$, instead of the hypothesis of Theorem 1.1.

- 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. $\Diamond$

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 $\mathcal{C}(n, d, s)$ will be used to denote the class of n-variate[4] polynomials of degree at most d that are computable by formulas of size at most s. $\Diamond$

We will use the following folklore algorithm for computing univariate polynomials, often attributed to Horner[5]. 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 $2d+1$. $\square$

The following observation shows that the classes of algebraic formulas/ABPs/circuits are *robust* under some very natural operations. These are precisely the properties of the underlying models that we rely on in this paper. Any circuit model that satisfies these properties would be sufficient for our purposes but we shall focus on just the standard models of formulas, ABPs and circuits.

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

[5] Though this method was discovered at least 800 years earlier by Iranian mathematician and astronomer Sharaf al-Dīn Ṭūsī (cf. Hogendijk [Hog89]).

Observation 2.4. *The class of polynomials computed by formulas/ABPs/circuits satisfy the following properties:*

- *Any polynomial of degree d with at most s monomials can be computed by a formula/ABP/circuit of size $s \cdot d$. In the specific setting when the polynomial is a univariate, it can be computed by a formula/ABP/circuit of size $O(d)$.*
- *Partial substitution of variables does not increase the size of the formula/ABP/circuit.*
- *If each of $Q_1, \ldots, Q_k$ is computable by size s formulas/ABPs/circuits, then $\sum Q_i$ is computable by size sk formula/ABP/circuit respectively.*
- *Suppose $P(x_1, \ldots, x_n)$ is computable by a size s_1 formula/ABP/circuit and say $Q_1, \ldots, Q_n$ are polynomials each of which can be computed by formulas/ABPs/circuits of size s_2. Then, $P(Q_1, \ldots, Q_n)$ can be computed by a formula/ABP/circuit of size at most $s_1 \cdot s_2$ respectively.* □

2.2 Combinatorial designs

Definition 2.5 (Nisan-Wigderson designs [NW94])**.** *A family of sets $\{S_1, \ldots, S_m\}$ is said to be an (ℓ, k, r) design if*

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

The following is a standard construction of such designs based on the *Reed-Solomon* code.

Lemma 2.6 (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* $\operatorname{poly}(m)$.

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 $|S_p \cap S_q| < r$ for $p \neq q$. □

2.3 Hardness-randomness connections

Observation 2.7. *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 2.8 (Hitting sets to hardness [HS80, Agr05])**.** *Let H be an explicit hitting set for $\mathcal{C}(n,d,s)$. Then, for any $k \leq n$ such that $k|H|^{1/k} \leq d$, there is a polynomial $Q(z_1,\ldots,z_k)$ of individual degree smaller than $|H|^{1/k}$ 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 $d' < |H|^{1/k}$, 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 be found by solving a system of linear equations in time $\mathrm{poly}(|H|)$. The degree of Q_k is at most $k \cdot |H|^{1/k} \leq d$ from the hypothesis and the hardness of Q_k follows from Observation 2.7. □

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 $Q(z_1,\ldots,z_k) \in \mathbb{F}[\mathbf{x}]$ 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 Implagliazzo [KI04] 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 [Kal89] (or even a non-algorithmic version due to Bürgisser [Bür00]) 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 or algebraic branching programs as we do not know if they are closed under factorization. The following lemma can be used in such settings and this paper makes heavy use of this.

Lemma 2.9 (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 2.7 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 factorization closure results.

Proof of Lemma 2.9. Borrowing the ideas from Kabanets and Impagliazzo [KI04], 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 $0 \le i \le m$, 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 $1 \le t \le m$, 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 $(s \cdot (r-1) \cdot d^r)$,
- $\deg(P') \le D \cdot \deg(Q) \le D \cdot (kd)$, and $\deg_{x_t}(P') \le D$,
- $P'(\mathbf{y}_t, Q(\mathbf{y}_t)) \equiv 0$.

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 $2d+1 \le 4d$. 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 et al. [AGS18] but with some simple but crucial 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 $\varepsilon > 0$ be a constant. For a large enough n, suppose that for all $s \geq n$ there is an explicit hitting set of size $s^{n-\varepsilon}$, for all degree s, size s algebraic formulas over n variables.*

Then for $m = n^8$ and for all $s \geq m$, there is an explicit hitting set of size $s^{m/50}$ 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 $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.*
2. *Suppose $g(n) \leq \left(\frac{1}{10}\right) \cdot n^{1/4}$, then for $m = 2^{n^{1/4}}$ 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 $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.

We will defer the proofs of these lemmas to the end of this section and complete the proof of Theorem 1.1.

Theorem 1.1 (Bootstrapping PIT for algebraic formulas, branching programs and circuits)**.** *Let $\varepsilon > 0$ be a constant. For some large enough n suppose that, for all $s \geq n$, there is an explicit hitting set of size $s^{n-\varepsilon}$ for all degree s, size s algebraic formulas (algebraic branching programs or circuits respectively) over n variables. Then, there is an explicit hitting set of size $s^{\exp\circ\exp(O(\log^* s))}$ for the class of degree s, size s algebraic formulas (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 $s^{n_0-\varepsilon}$ for $\mathcal{C}(n_0,s,s)$ and one application of Lemma 3.1 gives an explicit hitting set of size $s^{n_1/50}$ for $\mathcal{C}(n_1, s, s)$ for $n_1 = n_0^8$ and all $s \geq n_1$. Using Lemma 3.2(1) we obtain an explicit hitting set of size $s^{(1/10)\cdot m_0^{1/4}}$ 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 $m_i = 2^{m_{i-1}^{1/4}}$ 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 $t_0 = \left(\frac{1}{10}\right) m_0^{1/4}$ 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 $t_j = 20^{(2^j-1)} \cdot t_0^{2^j} = \exp \circ \exp(O(j))$.
- the exponent of s in the complexity of the final hitting set is $t_{O(\log^* s)} = \exp \circ \exp(O(\log^* s))$.

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

3.1 Proofs of the bootstrapping lemmas

Here we prove the two main lemmas used in the proof of Theorem 1.1. We restate the lemmas here for convenience. The proofs follow a very similar template but with different settings of parameters and minor adjustments.

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

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

Proof. Let $n > 150/\varepsilon$. We begin by fixing the design parameters, $k = n$, $\ell = k^5 = n^5$ and $r = 2$.

Constructing a suitably hard polynomial: For $B = 3k/\varepsilon$, 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 $s^{B(k-\varepsilon)}$ using Lemma 2.8. The polynomial $Q_k(\mathbf{z})$ has the following properties.

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

Building the NW design: Using Lemma 2.6, we now construct an (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m := k^8 = k^{(5-1)2}$.

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 2.9 (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

$$\begin{aligned}
\mathrm{size}(\tilde{P}) &\leq 4 \cdot s \cdot d \cdot (s+1) \\
&\leq 4s(s+1) \cdot s^{B(k-\varepsilon)/k} \\
&\leq s^{\left(3 + \frac{B(k-\varepsilon)}{k}\right)} = s^{3 + \frac{3k}{\varepsilon} - 3} = s^B.
\end{aligned}$$

Moreover, note that the degree of $\tilde{P}(z_1, \ldots, z_k)$ is at most $(k \cdot d) \cdot s \leq s^{\left(2 + \frac{B(k-\varepsilon)}{k}\right)} < s^B$. 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 2.7. 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 Ore-DeMillo-Lipton-Schwartz-Zippel lemma [Ore22, DL78, Zip79, Sch80], 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 $\frac{m}{50} = \frac{n^8}{50} > B\ell = \left(\frac{3n}{\varepsilon}\right) \cdot n^5$, because $n > 150/\varepsilon$.

The construction runs in time that is polynomial in the size of the hitting set in the conclusion. □

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 $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.*

2. *Suppose $g(n) \leq \left(\frac{1}{10}\right) \cdot n^{1/4}$, then for $m = 2^{n^{1/4}}$ 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 $h(m) \leq \left(\frac{1}{10}\right) \cdot m^{1/4}$.

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[6] $k \leq n$. This can be done by using Lemma 2.8. 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 2.6, 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 $P(Q_k[\![\ell, k, r]\!]_{\mathrm{NW}}) \equiv 0$, then Lemma 2.9 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

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

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 on $k \leq n$ 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 2.7 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 2.6, we have an explicit (ℓ, k, r) design $\{S_1, \ldots, S_m\}$ with $m = k^r = n^{10}$. 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 Ore-DeMillo-Lipton-Schwartz-Zippel lemma [Ore22, DL78, Sch80, Zip79], we have an explicit hitting set of size $s^{3\ell} \leq s^{3m^{1/5}}$. Since $m \geq n_0$ and n_0 is large enough, we have that

$$h(m) := 3m^{1/5} \leq \left(\frac{1}{10}\right) \cdot m^{1/4}.$$

Part (2) $\left(g(n) \leq \left(\frac{1}{10}\right) n^{1/4}\right)$**:** In this case, we choose $k = \sqrt{n}$ and $r = n^{1/4}$, so that $5r \cdot g(n) \leq 10r \cdot g(n) \leq k$ and $\ell = n$. Using Lemma 2.6, we now construct an explicit (ℓ, k, r) design

[6] 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.

$\{S_1, \ldots, S_m\}$ with $m = 2^{n^{1/4}} \leq k^r$.

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

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

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

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

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

4 Open problems

We end with some open questions.

- Can the conclusion in Theorem 1.1 be improved further? For instance, is it true that if we have explicit hitting sets of size $s^{n-\varepsilon}$, we can bootstrap these to get explicit hitting sets of size $s^{O(1)}$? As far as we understand, there does not seem to be a good reason for such a statement to be false.

- A natural question in the spirit of the results in this paper, and those in Agrawal et al. [AGS18] 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 arithmetic 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 et al. [CILM18] recently showed such results, but no such result appears to be known for commutative algebraic circuits.

- Kabanets and Impagliazzo [KI04] show that given a polynomial family which requires exponential sized circuits, there is a blackbox PIT algorithm which runs in 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 [KI04]. 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.

- And lastly, we would like to understand if it is possible to bootstrap white box PIT algorithms.

Acknowledgements: 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.

References

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2005.

[AGS18] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 1166–1179. ACM, 2018. `eccc:TR18-035`.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[BS83] Walter Baur and Volker Strassen. The Complexity of Partial Derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000.

[CILM18] Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness Amplification for Non-Commutative Arithmetic Circuits. In *Proceedings of the 33rd Annual Computational Complexity Conference (CCC 2018)*, volume 102 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. `eccc:TR18-095`.

[DL78] Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978.

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 754–763. ACM, 2016. `eccc:TR15-177`.

[For14] Michael Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, Massachusetts Institute of Technology, 2014.

[Gro] Joshua Grochow. `http://cstheory.stackexchange.com/questions/19261/degree-restriction-for-polynomials-in-mathsfvp/19268#19268`.

[Hog89] Jan P. Hogendijk. Sharaf al-Dīn Ṭūsī on the number of positive roots of cubic equations. *Historia Mathematica*, 16(1):69 – 85, 1989.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 262–272. ACM, 1980.

[Kal89] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS 1990)*, pages 2–10, 1990.

[MVV87] 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 the *19th Annual ACM Symposium on Theory of Computing (STOC 1987)*.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. Available on `citeseer:10.1.1.83.8416`.

[Ore22] Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922.

[Sap15] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015.

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980.

[Sha90] Adi Shamir. IP=PSPACE. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS 1990)*, pages 11–15, 1990.

[ST17] Ola Svensson and Jakub Tarnawski. The Matching Problem in General Graphs Is in Quasi-NC. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 696–707. IEEE Computer Society, 2017. `arXiv:1704.01929`.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A The algorithm for generating the hitting set

Let n_0 be the initial threshold, a large enough constant so that it satisfies the "large enough" requirements of Lemma 3.1 and Lemma 3.2.

$$\begin{aligned} & n_0 \text{ is large enough} && t_0 = (n_0 - \varepsilon) \\ & n_1 = n_0^8 && t_1 = {}^{n_1}\!/_{50} \\ & n_2 = n_1^{10} && t_2 = ({}^1\!/_{10})\, n_2^{1/4} \\ \text{For all } i \geq 3, \quad & n_i = 2^{n_{i-1}^{1/4}} && t_i := 20 t_{i-1}^2 \end{aligned}$$

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

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 $s^{\exp \circ \exp(O(\log^* s))}$. The overall running time is polynomial time in the size of the final hitting set which is $s^{t_b} = s^{\exp \circ \exp(O(\log^* s))}$.

Algorithm 1: HITTING-SET

```
Input  : Parameter $i$ and a size $s$.
Output: A hitting set of size $s^{t_i}$ size for $\mathcal{C}(n_i, s, s)$.
if $i = 1$ then
    Let $B = 3n_0/\varepsilon$ and $H_0(s^B) :=$ INITIAL-HITTING-SET$(s^B)$        // size at most $s^{B(n_0-\varepsilon)}$
    Compute a nonzero polynomial $Q$ on $k = n_0$ variables of individual degree smaller
      than $s^{Bt_0/k}$ that vanishes on $H_0(s^B)$.                  // takes poly$(s^{Bt_0})$ time
    Compute an $(n_0^5, n_0, 2)$-design $\{S_1, \ldots, S_{n_1}\}$.
    Let $S \subseteq \mathbb{F}$ be of size at least $s^B$.
    return $\left\{ (Q[\![\ell, k, r]\!]_{\mathrm{NW}}) (\mathbf{a}) \ : \ \mathbf{a} \in S^{n_0^5} \right\}$            // size at most $s^{Bn_0^5} \leq s^{n_1/50} = s^{t_1}$
else if $i = 2$ then
    $H_1(s^5) :=$ HITTING-SET$(1, s^5)$                              // size at most $s^{5t_1}$
    Compute a nonzero polynomial $Q$ on $k = n_1$ variables of individual degree smaller
      than $s^{5t_1/k}$ that vanishes on $H_1(s^5)$.                    // takes 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 $s^{3n_1^2} \leq s^{0.1 \cdot n_2^{1/4}} = s^{t_2}$
else if $i \geq 3$ then
    $H_{i-1}(s^5) :=$ HITTING-SET$(i-1, s^5)$                            // size at most $s^{5t_{i-1}}$
    Compute a nonzero polynomial $Q$ on $k = \sqrt{n_{i-1}}$ variables of individual degree smaller
      than $s^{5t_{i-1}/k}$ that vanishes on $H_{i-1}(s^5)$.               // takes 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 $s^{20t_{i-1}^2} = s^{t_i}$
```