Bootstrapping Variables in Algebraic Circuits[*]

Manindra Agrawal
IIT Kanpur
Kanpur, India
manindra@cse.iitk.ac.in

Sumanta Ghosh
IIT Kanpur
Kanpur, India
sumghosh@cse.iitk.ac.in

Nitin Saxena
IIT Kanpur
Kanpur, India
nitin@cse.iitk.ac.in

ABSTRACT

We show that for the blackbox polynomial identity testing (PIT) problem it suffices to study circuits that depend only on the first extremely few variables. One only need to consider size-s degree-s circuits that depend on the first $\log^{\circ c} s$ variables (where c is a constant and we are composing c logarithms). Thus, hitting-set generator (hsg) manifests a *bootstrapping* behavior— a partial hsg against very few variables can be efficiently grown to a complete hsg. A boolean analog, or a pseudorandom generator property of this type, is unheard-of. Our idea is to use the partial hsg and its annihilator polynomial to efficiently bootstrap the hsg exponentially wrt variables. This is repeated c times in an efficient way.

Pushing the envelope further we show that: **(1)** a quadratic-time blackbox PIT for 6913-variate degree-s size-s polynomials, will lead to a "near"-complete derandomization of PIT, and **(2)** a blackbox PIT for n-variate degree-s size-s circuits in $s^{n^{\delta}}$-time, for $\delta < 1/2$, will lead to a "near"-complete derandomization of PIT (in contrast, s^n-time is trivial).

Our second idea is to study depth-4 circuits that depend on constantly many variables. We show that a polynomial-time computable, $O(s^{1.49})$-degree hsg for *trivariate* depth-4 circuits bootstraps to a quasipolynomial time hsg for general poly-degree circuits, and implies a lower bound that is a bit stronger than Kabanets-Impagliazzo (STOC 2003).

CCS CONCEPTS

• **Theory of computation** → **Algebraic complexity theory**; *Fixed parameter tractability*; **Pseudorandomness and derandomization**; • **Computing methodologies** → **Algebraic algorithms**; • **Mathematics of computing** → Combinatoric problems;

KEYWORDS

hitting-set, tiny, depth-3, depth-4, derandomization, identity testing, lower bound, VP, VNP, E, #P/poly, SUBEXP, NW design, circuit factoring, approximative, tetration.

ACM Reference Format:
Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. 2018. Bootstrapping Variables in Algebraic Circuits. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3188745.3188762

[*]Full version here: https://www.cse.iitk.ac.in/users/nitin/papers/tiny-circuits.pdf.

1 INTRODUCTION

Polynomial identity testing (PIT) problem is to decide whether a multivariate polynomial is zero, where the input is given as an algebraic circuit. An algebraic circuit over a field $\mathbb{F}$ is a layered acyclic directed graph with one sink node called output node; source nodes are called input nodes and are labeled by variables or field constants; non-input nodes are labeled by $\times$ (multiplication gate) and $+$ (addition gate) in alternate layers. Sometimes edges may be labeled by field constants. The computation is defined in a natural way. The complexity parameters of a circuit are: 1) *size*- number of edges and vertices (including the *variables*), 2) *depth*- number of layers, and 3) *degree*- maximum degree among all polynomials computed at each node. Note– The degree of the computed polynomial may be much smaller than the degree of its circuit.

The polynomial computed by a circuit may have, in the worst-case, an exponential number of monomials compared to its size. So, by computing the explicit polynomial from input circuit, we cannot solve PIT problem in polynomial time. However, evaluation of the polynomial at a point can be done, in time polynomial in the circuit size, by assigning the values at input nodes. This helps us to get a polynomial time randomized algorithm for PIT by evaluating the circuit at a random point, since any nonzero polynomial evaluated at a random point gives a nonzero value with high probability [13, 64, 70]. However, finding a deterministic polynomial time algorithm for PIT is a long-standing open question in algebraic complexity theory. It naturally appears in the algebraic-geometry approaches to the P≠NP question, eg. [27, 28, 51, 52, 55]. The famous algebraic analog is the VP≠VNP question [67]. The PIT problem has applications both in proving circuit lower bounds [1, 35, 37] and in algorithm design [4, 16, 44, 53]. For more details on PIT, see the surveys [61, 62, 65] or review articles [54, 68].

PIT algorithms are of two kinds: 1) *whitebox*- use the internal structure of the circuit, and 2) *blackbox*- only evaluation of the circuit is allowed at points in a 'small' extension $\mathbb{K} \supseteq \mathbb{F}$. Blackbox PIT for a set of polynomials $\mathcal{P} \subset \mathbb{K}[\mathbf{x}]$ is equivalent to efficiently finding points $\mathcal{H} \subset \mathbb{K}^n$, called a *hitting-set*, such that for any nonzero $P \in \mathcal{P}$, the set $\mathcal{H}$ contains a point at which $P \neq 0$. For us a more functional approach would be convenient. We think in terms of an n-tuple of univariates $\mathbf{f}(y) = (f_1(y), \ldots, f_n(y))$, in $\mathbb{K}[y]$, whose set of evaluations contain an $\mathcal{H}$. Such an $\mathbf{f}(y)$ can be efficiently obtained from a given $\mathcal{H}$ (using interpolation) and vice-versa. Clearly, if $\mathcal{H}$ is a hitting-set for $\mathcal{P}$ then $P(\mathbf{f}(y)) \neq 0$, for any nonzero $P \in \mathcal{P}$. This tuple of univariates is called a hitting-set generator (*hsg*) and its *degree* is $\max_{i \in [n]} \deg(f_i)$, which is $\leq |\mathcal{H}|$.

Our Work. We study the phenomenon of *bootstrapping*: converting an hsg for size-s degree-s n-variate circuits to hsg for size-s

degree-s $L(n)$-variate circuits with $L(n) > n$. In the boolean settings, this phenomenon is well understood. The analog of hsg is *pseudo-random generator (prg)* that *stretches a seed* by several bits, or, the *s-extender* that stretches n by a *single* bit. By [56, Sec.2-3] it is known that an extender for size-s $(\log s)$-variate boolean circuits can be converted to an optimal prg for size-s circuits with $L(n) = 2^n$. No further "reduction" in number of variables is possible since the size of a $(\epsilon \log s)$-variate circuit can be reduced to $< s$ if $\epsilon < 1$.

The situation is less clear in algebraic settings. On one hand, n-variate polynomials requiring circuits of size s exist for every n and s (due to the fact that polynomials can have arbitrarily large degrees unlike boolean settings where every function is multilinear). On the other hand, bootstrapping from $O(\log s)$ variables to s variables is not studied explicitly in the literature.

We close this gap in knowledge by showing that an hsg for size-s degree-s $(\log^{\circ c} s)$-variate circuit can be efficiently converted to an hsg for size-s degree-s s-variate circuit; where $\log^{\circ c} s :=$ $\log \cdots (c\ times) \cdots \log s$. Furthermore, at the cost of making the final hsg slightly superpolynomial $= s^{\exp\circ\exp(O(\log^\star s))}$, we show that bootstrapping can be done from even a *constant* number of variables! Our results can also be viewed as a powerful *amplification* of derandomization: a "slight" derandomization ($= s^{n^\delta}$ time hsg for size-s degree-s n-variate circuits, for a constant $\delta < 1/2$) implies "nearly" complete derandomization ($= s^{\exp\circ\exp(O(\log^\star s))}$ time hsg for size-s degree-s s-variate circuits). Compare the required s^{n^δ}-time PIT with the trivial s^n-time PIT.

We prove an additional result for shallow circuits: poly(s)-time computable and $O\left(s^{n/2}/\log^2 s\right)$ degree hsg for size-s n-variate depth four circuits (for some constant $n \geq 3$) implies quasipolynomial time blackbox PIT for size-s degree-s s-variate circuits (& strong exponential lower bounds). See Theorems 1–4 for more formal statements.

We see our results as a positive development; since, they reduce PIT to cases that are special in an unprecedented way. Such special-case PIT algorithms are waiting to be discovered.

Existing deterministic algorithms solving PIT for restricted classes have been developed by leveraging insight into their weaknesses. For example, deterministic PIT algorithms are known for subclasses of depth-3 circuits [40, 60, 63], subclasses of depth-4 circuits [5, 9, 19, 45, 46, 57, 58], read-once algebraic branching programs (ROABP) and related models [3, 6, 21, 22, 31, 32, 50], certain symbolic determinants [17, 33, 34, 66], as well as non-commutative models [26, 47, 48]. An equally large number of special models have been used to prove lower bounds, see for example the ongoing online survey of Saptharishi [59]. Also, blackbox PIT relates to conjectures that bar certain algebraic circuit lower bound methods [24].

Our Notation. $[n]$ refers to $\{1, 2, \ldots, n\}$. Logarithms are wrt base 2. Iterated logarithm $\log^\star s$ is the least number of iterated applications of log that gives a result ≤ 1. When we say that a circuit is of size-s (resp. depth-Δ, or degree-d) we use the parameters as an *upper* bound.

Field. To appreciate the most important aspects of this work keep in mind the "practical" fields $\mathbb{F} = \mathbb{Q}$ or $\mathbb{F}_q$. Interestingly, our main theorems (Thms. 1–4) hold for *any* field. However, the other theorems require field characteristic to be zero or large. Common examples are: complex $\mathbb{C}$, reals $\mathbb{R}$, algebraic numbers $\overline{\mathbb{Q}}$, local fields $\mathbb{Q}_p$ or their extensions, or finite fields $\mathbb{F}_q$ of characteristic $p >$ degree of the input.

Finally, one can generalize our work to the field $K = \mathbb{F}(\epsilon)$ with $\epsilon \to 0$ in a certain way. This leads to *approximative* complexity $\overline{\text{size}}$ of polynomials in $\mathbb{F}[\mathbf{x}]$ [10, Defn.3.1]. Efficient hitting-sets wrt $\overline{\text{size}}$ are equivalent to explicit system of parameters (*esop*) of the invariant ring of a related variety $\Delta[\det(X), s]$ with a given group action [52, Thm.4.9]. Our work (Theorem 4) will imply that to prove the existence of such a (quasi-)esop it suffices to study esop wrt X that depend on 'constantly few' variables (also see the reduction of derandomized Noether Normalization problem NNL to blackbox PIT in [52, Sec.4.3]).

A basic algebraic algorithm used in our results is circuit factoring, that relies on field properties. A classic result is [39] that constructs small circuits for factors that have multiplicity coprime to the characteristic (see [15] for recent factoring results and the related rich background).

Hitting-set Generator (hsg). Let $\mathcal{P}$ be a set of n-variate polynomials. We call an n-tuple of univariates $\mathbf{f}(y) = (f_1(y), \ldots, f_n(y))$ a (t, d)*-hsg* for $\mathcal{P}$ if: (1) for any nonzero $P \in \mathcal{P}$, $P(\mathbf{f}(y)) \neq 0$, and (2) $\mathbf{f}$ has time-complexity t and the degree of each f_i is less than d. By *t-time hsg* or *t-time hitting-set* or *t-time blackbox PIT*, we always mean a (t, t)-hsg.

The computational problem of designing and verifying an hsg for size-s circuits family is in PSPACE; however, that for $\overline{\text{size}}$-s circuits family is in EXPSPACE (recently brought down to PSPACE [23, 29]). The major open question is to bring this complexity down to P; this is christened 'GCT Chasm' in [52, Sec.11] and has since then become a fundamental difficulty common to geometry and complexity theories. It means that we have to discover algebraic properties that are specific to only those polynomials that have *small* circuit representation. We will investigate such properties closely in this work.

Variables. A polynomial P computed by a size-s algebraic circuit C can have at most $\{x_1, \ldots, x_s\}$ variables. For $k < s$, if we say that C *depends only on the first* k *variables*, then it is meant that the computed polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_k]$.

Multi-δ-ic. A polynomial family $\{f_n(x_1, \ldots, x_n)\}_{n\geq 1}$ over a field $\mathbb{F}$ is called *multi-δ-ic*, if degree of each variable in f_n is less than δ. For eg. when $\delta = 2$, $\{f_n\}_{n\geq 1}$ is multilinear.

E-computable Polynomial Family. For constant δ, a multi-δ-ic polynomial family $\{f_n\}_n$ with integer coefficients is called *E-computable* if: there exists a $2^{O(n)}$-time algorithm that on input $\mathbf{e}$, outputs the coefficient of $\mathbf{x}^{\mathbf{e}}$ in f_n in binary; say the leading bit will denote the *sign* of the coefficient, with 0 implying a positive coefficient and 1 implying negative. This makes coeff$(\cdot)(f_n)$ a boolean function ($\{0,1\}^* \to \{0,1\}^*$) whose bits are E-computable as well.

1.1 Our Motivation and Main Results

Pseudorandom generator (*prg*) is a well studied object in boolean circuit complexity theory and cryptography [69] & [8, Chap.10]. One of the main motivations of studying prg is to efficiently derandomize all randomized algorithms. Indeed one can show that if we have an *optimal prg* against BPP, then BPP=P. By optimal prg, we mean a prg which stretches an n-length string to 2^n-length and is computable in $2^{O(n)}$ time. Interestingly, an optimal prg is closely related to strong circuit lower bound. It is a celebrated result that designing optimal prg against P/poly is equivalent to finding an E-computable boolean function which has boolean circuit complexity $2^{\Omega(n)}$ [56, Secs.2.5 & 3.1], [36, Thm.2].

Naturally, an algebraic analog of the latter property would be to identify an E-computable polynomial family which has *algebraic* circuit complexity $2^{\Omega(n)}$. By Valiant's criterion [11, Prop.2.20] if one replaces E by #P/poly then we are directly talking about a *strong* version of VNP≠VP. As a first challenge, we can pose the following reasonable complexity conjecture.

Conjecture 1. There is an E-computable polynomial which has algebraic complexity $2^{\Omega(n)}$. Thus, either $E \not\subseteq$ #P/poly or VNP has a polynomial family of algebraic circuit complexity $2^{\Omega(n)}$.

In the world of algebraic circuits, hitting-set generator (hsg) is in direct analogy with prg. So one can naturally ask about the relation between hsg and algebraic circuit lower bound. Heintz and Schnorr [35, Thm.4.5] introduced the concept of an efficient annihilator of the hsg. They showed that if we can efficiently compute an hsg for a set of polynomials $\mathcal{P}$, then we can also efficiently compute a polynomial (namely, annihilator) which does not belong to $\mathcal{P}$. This technique can be easily extended to get the following circuit lower bound result. Like boolean world, our hard polynomial is also E-computable but has algebraic circuit complexity $2^{\Omega(n)}$.

Theorem 0 (Connection). If we have poly(s)-time blackbox PIT for size-s degree-s circuits $\mathcal{P}_s$, then Conjecture 1 holds. (Proof sketched in Section A.)

A weak converse of the above theorem, i.e. hardness to hsg, is well-known due to [38, Thm.7.7]. We state a revised version of it as Lemma 9. If we have an exponentially hard but E-computable polynomial family, then by using Lemma 9 we can efficiently reduce the number of variables in any circuit, from s to $O(\log s)$, preserving the nonzeroness. Next, one applies a "trivial" hitting-set on the $O(\log s)$ variables, which gives a quasipolynomial time hsg for $\mathcal{P}_s$ [12]. This suggests that the 'hardness vs randomness' connection here is less satisfactory than the boolean world. Nonetheless, one wonders whether the conclusion in Theorem 0 can be strengthened in a different way, so that we get a perfect equivalence. In this work, we answer this question by introducing the concept of partial hsg. Indeed, we give infinitely many different-looking statements that are all equivalent to the hypothesis in Theorem 0.

Partial Hsg. For all $s \in \mathbb{N}$, let $\mathbf{g}_s = (g_{s,1}(y), \ldots, g_{s,s}(y))$ be an hsg of $\mathcal{P}_s$. Suppose we can efficiently compute only the first "few" polynomials of the hsg. Can we *bootstrap* it, i.e. recover the whole hsg efficiently? Formally, we can describe this as follows. For any $m \in [s-1]$, the partial hsg $\mathbf{g}_{s,m}$ is defined as $(g_{s,1}, \ldots, g_{s,m})$. The partial hsg $\mathbf{g}_{s,m}$ can be seen as the hsg of those polynomials in $\mathcal{P}_s$ which depend only on the first m variables. Suppose that for $m \ll s$, we can compute $\mathbf{g}_{s,m}$ in poly(s)-time. Then, using this partial hsg, can we also design a complete hsg for $\mathcal{P}_s$ in poly(s)-time?

If $m = s^{1/c}$ for some $c \in \mathbb{N}$, then the answer is 'Yes' and it follows from the definition. The set $\mathcal{P}_s$ can be thought of as a subset of those polynomials in $\mathcal{P}_{s^c}$ which depend only on the first s variables. So $\mathbf{g}_{s^c,s} = (g_{s^c,1}, \ldots, g_{s^c,s})$ is a hsg for $\mathcal{P}_s$. Clearly, $\mathbf{g}_{s^c,s}$ can be computed in poly(s)-time. However, for $m \leq s^{o(1)}$, we cannot use the same argument for the following reason. To compute the hsg of $\mathcal{P}_s$, we have to compute the partial hsg for $\mathcal{P}_{s^{\omega(1)}}$, which may not be computable in poly(s)-time. Naively speaking, there is no reason why a partial hsg $\mathbf{g}_{s,s^{o(1)}}$ could be bootstrapped efficiently to $\mathbf{g}_s$. The former is a property of the polynomial ring $\mathbb{F}[x_1, \ldots, x_{s^{o(1)}}]$ compared to that of the latter "much larger" polynomial ring $\mathbb{F}[x_1, \ldots, x_s]$; so in the underlying algebraic-geometry concepts a terrible blow up is warranted.

For any $c \in \mathbb{N}$, let $\log^{\circ c}$ be defined as c-times application of the base-2 logarithm function (eg. $\log^{\circ 3} s = \log\log\log s$). Somewhat surprisingly, we give a positive answer for m as small as $\log^{\circ c} s$, for any $c \in \mathbb{N}$. For smaller values of m (eg. $m = \log^{\star} s$), we leave it as an open question.

Theorem 1 (Bootstrap hsg). *Suppose, for some $c \in \mathbb{N}$, we have a poly(s)-time blackbox PIT for size-s degree-s circuits that depend only on the first $\lceil \log^{\circ c} s \rceil$ variables. Then, we have a poly(sd)-time blackbox PIT for size-s degree-d circuits and Conjecture 1 holds.*

Remark. 1) In the boolean world, there is no extender that can stretch $0.99 \log s$ bits and "fool" size-s circuits. Because boolean functions on that many bits have circuit-size $< s$.

2) We also study the case when our partial hsg can be computed in subexponential time, which is far worse than polynomial time. In this case, our result is not as strong as Theorem 1. However, in the hypothesis we still deal with an $m = s^{o(1)}$ and manage to bootstrap that partial hsg in subexponential time. Also, an E-computable super-polynomially hard polynomial family is implied (say, *weak Conjecture 1*). For details see Theorem 13.

The bootstrapping idea brings forth pleasant surprises if we are willing to content ourselves with a "slightly super"-polynomial time blackbox PIT in the conclusion. Though we do not get an equivalence result now, we do however weaken the hypothesis very significantly.

Theorem 2. *Suppose, for constants $e \geq 2$ and $1 > \epsilon \geq (3 + 6\log(128e^2))/(128e^2)$, we have an $O(s^e)$-time blackbox PIT for degree-s polynomials computed by size-s circuits that depend only on the first $n := \lceil \max\{192e^2 \log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\} \rceil$ variables. Then, we have an $s^{\exp\circ\exp(O(\log^{\star} s))}$-time blackbox PIT for size-s degree-s circuits and Conjecture 1 holds.*

Remark. If we fix $e = 2$ and $\epsilon = 6912/6913$, then the hypothesis required is: Quadratic-time (i.e. $O(s^2)$) blackbox PIT for 6913-variate degree-s size-s polynomials.

In the above theorem, the exponent e in the complexity of PIT is a constant just below $\sqrt{n}/8$, where n is the (constant) number of variables. This can be achieved from a "poor" quality blackbox PIT algorithm (varying both s and n as independent parameters):

Theorem 3. *Suppose, for constant $\delta < 1/2$, we have an $s^{n^{\delta}}$-time blackbox PIT for size-s degree-s circuits that depend only on the first n variables. Then, we have an $s^{\exp\circ\exp(O(\log^{\star} s))}$-time blackbox PIT for size-s degree-s circuits and Conjecture 1 holds.*

Note that in an n-variate degree-s polynomial, there are at most $1 + s^n$ monomials. So, the above hypothesis is unexpectedly weak.

Additionally, the lower bound result that it will give is truly exponential. Next, we show that bootstrapping can be done even at shallow depths.

Theorem 4 (Depth-4 tiny variables). *Suppose, for constant $n \geq 3$, we have a $\left(poly(s^n), O\left(\frac{s^{n/2}}{\log^2 s}\right)\right)$-hsg for size-$s$ depth-4 circuits that depend only on the first n variables. Then, we have a quasipoly(sd)-time blackbox PIT for size-s, degree-d circuits and Conjecture 1 holds.*

Remark. 1) If we fix $n = 3$, then the hypothesis required is: $\left(\text{poly}(s), O(s^{1.5}/\log^2 s)\right)$-hsg for trivariate size-s depth-4 circuits. While $\left(\tilde{O}(s^3), (s+1)^3\right)$-hsg is trivial to design.

2) Depth-4 circuit is denoted as $\Sigma\Pi\Sigma\Pi$ to specify the alternating layers starting with the top addition gate. In older works it had been fruitful to restrict one of the product layer to mere *powering* gates [30, 60]. Indeed, we can prove stronger versions of Theorem 4: for $\Sigma\wedge\Sigma\Pi$ (Theorem 20) resp. $\Sigma\Pi\Sigma\wedge$ (Theorem 22) circuits in the hypothesis. But, these results (unlike Theorems 1–4) require the field characteristic to be zero or large.

3) Our conclusion is as strong as those obtained via the well-known 'constant-depth reduction' results in [7, 30]. But our hypothesis needs an hsg only slightly better than the trivial; this cannot be done, not even guessed, using the old methods.

Finally, we want to change the viewpoint and see blackbox PIT for depth-3 circuits through the lens of *fixed parameter tractability* (fpt). This is discussed in Section 5.1. Bootstrapping of variables from *log*-variate width-2 ABP is done in Section 5.2.

1.2 Proof Idea and Our Techniques

Proof Idea of Theorem 1. We have to prove two results; one related to PIT and the other one related to lower bound. The latter will follow from Theorem 0, so we only describe the proof idea of PIT part. Suppose that for all $s, d, i \in \mathbb{N}$, $\mathcal{P}_{s,d,i}$ is the set of degree-d polynomials computed by size-s circuits that depend only on the first $f_i(sd)$ variables, where $f_i(s)$ is intended to be $\omega(\log^{\circ i} s)$. For all $0 \leq i \leq c+1$, $f_i(s) := (\log^{\circ i} s)^2$. Using reverse induction, we show that for $0 \leq i \leq c+1$, we have a poly(sd) time hsg for $\mathcal{P}_{s,d,i}$. First, we design a poly(sd) time hsg for $\mathcal{P}_{s,d,c+1}$ using the hypothesis mentioned in the theorem. Next, for all $i \in [c+1]$, we use the poly$(s'd')$ time hsg of $\mathcal{P}_{s',d',i}$ to design a poly(sd) time hitting-set of $\mathcal{P}_{s,d,i-1}$.

Our induction step can be broken into three smaller steps.

1) Hsg of $\mathcal{P}_{s',d',i}$ to hard polynomial family: For all $s \in \mathbb{N}$, let $\mathcal{T}_{s,i}$ be the s-degree polynomials computed by size-s circuit that depends only on the first $2c_1\lceil\log^{\circ i} s\rceil$ variables, where c_1 is some constant. Using poly$(s'd')$ hsg of $\mathcal{P}_{s',d',i}$, we can design a poly(s) time hsg for $\mathcal{T}_{s,i}$. Applying Lemma 5, we consider an annihilator, of the hsg, and get a family of hard polynomials which satisfies the properties mentioned in Lemma 12 (that we need in the next step).

2) Hard polynomial to variable reduction map: Lemma 12 designs an efficient variable reduction map using a hard polynomial family with certain properties. Thus, we perform a variable reduction on the polynomials in $\mathcal{P}_{s,d,i-1}$; significantly reducing variables from f_{i-1} to f_i.

3) The map to poly(sd) time hsg for $\mathcal{P}_{s,d,i-1}$: The above variable reduction converts every nonzero polynomial in $\mathcal{P}_{s,d,i-1}$ to a nonzero one in $\mathcal{P}_{s',d',i}$, where $s', d' = \text{poly}(sd)$. Thus, on applying the polynomial time hsg for $\mathcal{P}_{s',d',i}$, we get a polynomial time hsg for $\mathcal{P}_{s,d,i-1}$.

The crucial technical step is provided by Lemma 12, which is a strict generalization of Lemma 9. As mentioned earlier, the latter itself is a revised version of [38, Thm.7.7] as it can handle hard *non*-multilinear polynomials. It designs an efficient variable reduction using an exponentially hard but E-computable polynomial family. If we have a poly(s) time hsg for $\mathcal{T}_{s,1}$, then using Lemma 5, one can get such a polynomial family (as in the proof of Theorem 0 but now the hard polynomial will be non-multilinear). In Step 1 above, we are working with poly(s) time hsg for $\mathcal{T}_{s,i}$, where $i > 1$. In such an extremely low variate regime, Lemma 5 cannot give us a polynomial family with constant individual degree. So, we cannot use Lemma 9 ideas if we desire polynomial time computation.

There are several technical challenges faced in choosing parameters that should lead to a contradiction in the proof. Since the individual degree of the hard polynomial depends on the time-complexity s^e of the hsg of $\mathcal{T}_{s,i}$, the factor circuits will have a blown up size after using Kaltofen factoring. Care is needed to counterbalance the size of the Nisan-Wigderson (NW) design and the hardness of the polynomial family with the circuit complexity of the factors. The more sophisticated statement of Lemma 12 takes care of all those issues. Why is this lemma invoked multiple times? The answer lies in the way Kaltofen factoring yields a contradiction: using the fact that the third-parameter (i.e. set-intersection size) in the NW design is much smaller than the second-parameter (i.e. set size). This gives a smaller factor of the composite circuit after fixing certain variables. So, we need to apply NW design for *each* exponential stretch of variables; we do not know how to directly get a *hyper*-exponential stretch and save on time.

Proof Idea of Theorem 2. Theorem 1 assumes an s^e-time hsg, where e is a constant, for $\log^{\circ c} s$-variate degree-s size-s polynomials. On the other hand, Theorem 2 assumes an s^e-time hsg for n-variate degree-s size-s polynomials, where $n := \lceil\max\{192e^2\log(128e^2)^{1/\epsilon}, (64e^2)^{1/\epsilon}\}\rceil$ and $1 > \epsilon \geq (3 + 6\log(128e^2))/(128e^2)$ are constants. In both the cases, our hypotheses demand improved hsgs over the trivial ones (namely, $s^{\log^{\circ c} s}$ and s^n time respectively). This is the common strength of both the hypotheses which is exploited in the proofs.

Broadly, the proof of Theorem 2 is similar to the previous one. However, in Theorem 2 we desire, for a given e, to find the minimum number of *constant* variables for which we can reach the conclusion. This imposes more technical challenges and in many steps of the proof we have to work with much finer parameters. For example, our calculation suggests that for $e = 2$, the number of variables that we need is $n = 6913$ (or, for $e = 3$, $n = 17574$ suffices).

Like Theorem 1, in each inductive step, we stretch the number of variables exponentially. However, here we finally stretch n variables to s variables, where n is a constant. So, we need around $\log^\star s$ steps, which is non-constant wrt s. We show that if we have an s^{f_i}-time hsg, in the i-th induction step, then in the next step we get an $s^{f_{i+1}}$-time hsg, where $f_{i+1} := 16f_i^2$. So, after $\log^\star s$ steps, we get an hsg of our desired complexity (=slightly super-polynomial).

Like Lemma 12, here Lemma 18 combines all the crucial tools needed in the inductive step of Theorem 2. Our key ingredients here are again Nisan-Wigderson design and Kaltofen's factoring. However, we use them in a more optimized way. It will help us to improve the constants that underlie. This theorem and the next are very sensitive to these technicalities.

Thus, we show that a significant improvement of blackbox PIT within the polynomial-time domain itself (from s^n to s^e) would near-completely solve PIT for VP. This reminds us of other famous algebraic problems in computing where improvements in the time-exponent have been widely studied (& still open)— integer multiplication [25] and matrix multiplication [49].

Proof Idea of Theorem 3. Suppose we have, for constant $\delta < 1/2$, an s^{n^δ}-time hsg for size-s degree-s circuits that depend only on the first n variables. Then, there exists an $\epsilon \in [2\delta, 1)$ and a large enough *constant* e such that: there is an s^e-time hsg for size-s degree-s circuits that depend only on the first $n := \lceil (64e^2)^{1/\epsilon} \rceil \geq 192e^2 \log(128e^2)^{1/\epsilon}$ variables. Note that $e \geq (n-1)^{\epsilon/2}/8 > n^\delta$ can be easily ensured, thus, s^e-time is more than s^{n^δ}-time. Now we simply invoke Theorem 2.

In fact, this proof needs the hypothesis only for: *infinitely* many n and large enough s.

Proof Idea of Theorem 4. We argue using two intermediate models. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of polynomials computed by size-s $\Sigma \wedge^a \Sigma\Pi$ circuits, $a(s)$ is an arbitrarily slow growing function, that depend only on the first n variables. Let $\mathcal{T}_s$ be the set of polynomials computed by size-s $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables.

To prove Theorem 4, first we show that $(\text{poly}(s), O(s^{n/2}/\log^2 s))$-hsg for $\mathcal{P}_s$ resp. $\mathcal{T}_s$ gives an efficient variable reduction and Conjecture 1 (see Theorems 20 resp. 22). This variable reduction converts a d-degree *nonzero* polynomial computed by a size-s circuit to a $O(\log(sd))$-variate poly(sd)-degree *nonzero* polynomial. For $O(\log(sd))$-variate and poly(sd)-degree polynomials, we have a $(sd)^{O(\log(sd))}$ time hitting-set. This completes the proof of PIT part. Next, we give the proof sketch of the variable reduction part.

First, we discuss the variable reduction part assuming $\mathcal{P}_s$ has $O(s^{n/2}/\log^2 s)$-degree hsg. We do it via an intermediate *multilinear* model. For all $s \in \mathbb{N}$, let $\mathcal{P}'_s$ be the set of $\frac{n}{2}\log s$ degree multilinear polynomials computed by size-s $\Sigma \wedge^a \Sigma\Pi$ circuits, that depend only on the first $n \log s$ variables. Next we describe how to get a hard polynomial family from an $(\text{poly}(s), O(s^n/\log^2 s))$-hsg of $\mathcal{P}'_s$.

The number of $\frac{n}{2}\log s$ degree multilinear monomials over $m := n\log s$ variables is $\binom{m}{m/2} \geq 2^m/\sqrt{2m} = s^n/\sqrt{2m}$, which is greater than $O(s^n/\log^2 s)\cdot m$ (for large enough s). So, we get an m-variate and $(m/2)$-degree multilinear homogeneous polynomial (annihilator) $q_m \notin \mathcal{P}'_s$ and computable in poly(s) time. The linear algebra is similar to Lemma 5; only difference being that Lemma 5 does not ensure q_m multilinear. However, the parameters of $\mathcal{P}'_s$ ensure the latter. Since q_m is m-variate $(m/2)$-degree multilinear polynomial and is not in $\mathcal{P}'_s$, q_m is not computed by size-s $\Sigma \wedge^a \Sigma\Pi$ circuits. Using depth reduction of [7], one can also ensure that q_m has circuit complexity $> s \geq 2^{\Omega(m)}$. This in turn gives the variable reduction using Lemma 9.

Now we show that an efficient $O(s'^{n/2}/\log^2 s')$-degree hsg of $\mathcal{P}_{s'}$ gives an efficient $O(s^n/\log^2 s)$-degree hsg for $\mathcal{P}'_s$, where s and s' are polynomially related. In $\mathcal{P}'_s$, divide the $n\log s$ variables into n blocks with each block of length $\log s$. Now take fresh variables $y_1, \ldots, y_n$, one for each block, and apply Kronecker map $(x_{u(j)+i} \mapsto y_j^{2^i}, i \in [\log s])$ within the j-th block $\{x_{u(j)+i} | i \in [\log s]\}$. Since polynomials in $\mathcal{P}'_s$ are multilinear, the above map preserves nonzeroness. This converts a nonzero polynomial in $\mathcal{P}'_s$ to a nonzero polynomial in $\mathcal{P}_{s'}$, where $s' = O(s^2)$. Now use the $O(s'^{n/2}/\log^2 s')$-degree hsg of $\mathcal{P}_{s'}$ to get one for $\mathcal{P}'_s$. For details see the proof of Theorem 20.

Second, we discuss the variable reduction part assuming an efficient $O(s^{n/2}/\log^2 s)$-degree hsg of $\mathcal{T}_s$. Proof idea is similar to the previous one; only difference is in the intermediate model. Here we consider the following model: for all $s \in \mathbb{N}$, let $\mathcal{T}_s{}'$ be the set of *multilinear* polynomials computed by size-s $\Sigma\Pi\Sigma$ circuits that depend only on the first $n\log s$ variables. Again, we show that an efficient $O(s'^{n/2}/\log^2 s')$-degree hsg of $\mathcal{T}_{s'}$ gives an $O(s^n/\log^2 s)$-degree hsg for $\mathcal{T}_s{}'$, which in turn gives the variable reduction as above coupled with [30]. For details see Theorems 21 & 22.

2 BRUSHING-UP RELEVANT TECHNIQUES

In this section we will revisit the techniques that have appeared in some form in [1, 7, 30, 35, 37, 56].

From a hitting-set generator $\mathbf{f}(y)$ of a set of polynomials $\mathcal{P}$, we get an explicit polynomial outside $\mathcal{P}$ simply by looking at an annihilating polynomial of $\mathbf{f}(y)$. Previously, this approach was discussed in [35, Theorem 4.5] and [1, Theorem 51]. In the next lemma, we prove a revised version. Later, it will be used to get hard polynomial from hitting-set generator.

Lemma 5 (hsg to hardness). *Let $\mathbf{f}(y) = (f_1(y), \ldots, f_n(y))$ be a (t,d)-hsg for a set of n-variate polynomial $\mathcal{P}$. Then, there exists an n-variate polynomial $g(\mathbf{x})$ that is not in $\mathcal{P}$, is computable in* poly(tdn)*-time, has individual degree less than $\delta := \lceil d^{3/n} \rceil$, and is homogeneous of degree $(\delta-1)n/2$.*

Proof. A natural candidate for $g(\mathbf{x})$ is any annihilating polynomial of the n polynomials $\mathbf{f}(y) = (f_1(y), \ldots, f_n(y))$, since for every nonzero $h \in \mathcal{P}$, $h(\mathbf{f})$ is nonzero. Define $\delta \geq 2$ as the smallest integer such that $\delta^{n/3} > d$. Consider $g(\mathbf{x})$ as a n-variate polynomial with individual degree less than δ and homogeneous of degree $(\delta-1)n/2$. Then, $g(\mathbf{x})$ can be written as:

$$g(\mathbf{x}) = \sum_{|\mathbf{e}|=(\delta-1)n/2\ ,\ 0\leq \mathbf{e}_i<\delta} c_{\mathbf{e}}\mathbf{x}^{\mathbf{e}} \qquad (1)$$

where, $c_{\mathbf{e}}$'s are unknown to us. Note that the number of summands is at least $(\delta/2)^{n/2}\cdot\binom{n}{n/2} > \delta^{n/2}$ (for $n \geq 4$). The former estimate can be obtained by picking a subset $S \in \binom{[n]}{n/2}$ and considering all monomials in $\mathbf{x}_S$ of individual-degree $< \delta/2$. For every such monomial in $\mathbf{x}_S$ we can pick a (complementary) monomial $\mathbf{x}_{[n]\setminus S}$ with exponents from $\{\delta/2, \ldots, \delta-1\}$ such that the product of these two monomials has degree *exactly* $(\delta-1)n/2$.

We can fix all the $c_{\mathbf{e}}$'s to zero except the ones corresponding to an index-set I of size $\delta_0 := dn(\delta-1)/2+2 < \delta^{n/3}n(\delta-1)/2+2 \leq \delta^{n/2}$. This way we have exactly δ_0 unknown $c_{\mathbf{e}}$'s. To be an annihilating polynomial of $\mathbf{f}(y)$, we need $g(\mathbf{f}) = 0$. By comparing the coefficients of the monomials in y, both sides of Equation 1, we get a linear system in the unknowns.

Suppose that δ_1 is the degree of y in $g(\mathbf{f})$. Then, $g(\mathbf{f})$ can be written as $g(\mathbf{f}) = \sum_{i=0}^{\delta_1} p_i \cdot y^i$, where p_i's are linear polynomials in $c_{\mathbf{e}}$'s. The constraint $g(\mathbf{f}) = 0$ gives us a system of linear equations with the number of unknowns δ_0 and the number of equations $\delta_1 + 1$. The value of δ_1 can be at most $d \cdot n \cdot (\delta - 1)/2$, which means that the number of unknowns δ_0 is greater than the number of equations δ_1. So, our system of *homogeneous* linear equations always has a nontrivial solution, which gives us a nonzero g as promised.

Computing $\mathbf{f}(y)$ takes t time and a solution of the linear equations can be computed in poly(tdn)-time. So, $g(\mathbf{x})$ can be computed in poly(tdn)-time. □

Corollary 6 (E-computable). *In the proof of Lemma 5, if $td = 2^{O(n)}$ then the polynomial family $g_n := g$, indexed by the variables, is E-computable.*

Proof. The linear system that we got can be solved in poly(tdn)-time. As it is homogeneous we can even get an integral solution in the same time-complexity. Thus, assuming $td = 2^{O(n)}$, the time-complexity of computing coef$_{\mathbf{x}^{\mathbf{e}}}(g)$ is poly(tdn)=poly(2^n) and g is multi-δ-ic ($\because$ $\delta = \lceil d^{3/n} \rceil = O(1)$). In other words, if we consider the polynomials $g_n := g$, indexed by the variables, then the family $\{g_n\}_n$ is E-computable. □

Towards a converse of the above lemma, a crucial ingredient is the Nisan-Wigderson design [56]. To describe it simply, the design *stretches a seed* from ℓ to $m \geq 2^{\frac{d}{10}}$ as follows,

Definition 7. *Let $\ell > n > d$. A family of subsets $\mathcal{D} = \{I_1, \ldots, I_m\}$ on $[\ell]$ is called an (ℓ, n, d)-design, if $|I_i| = n$ and for all $i \neq j \in [m]$, $|I_i \cap I_j| \leq d$.*

Lemma 8 (Nisan-Wigderson design, Chap.16 [8]). *There exists an algorithm which takes (ℓ, n, d) and a base set S of size $\ell > 10n^2/d$ as input, and outputs an (ℓ, n, d)-design $\mathcal{D}$ having $\geq 2^{d/10}$ subsets, in time $2^{O(\ell)}$.* (Lemma 14 improves this.)

Our next lemma is a revised version of the counterpositive of [38, Lemma 7.6]. If we have an exponentially hard but E-computable polynomial family, then we can efficiently reduce the variables from n to $O(\log(sd))$, for n-variate d-degree polynomials computed by size-s circuits, preserving nonzeroness. For proof, see the full version linked on the first page.

Lemma 9 (Hardness to variable reduction). *For some constant δ, let $\{q_m\}_{m \geq 1}$ be a multi-δ-ic polynomial family computable in $\delta^{O(m)}$ time, but it has no $\delta^{o(m)}$-size algebraic circuit.*

Then, for n-variate d-degree polynomials computed by size-s circuits we have a $\delta^{O(\log(sd))}$-time variable-reducing polynomial map, from n to $O(\log(sd))$, that preserves nonzeroness. Furthermore, after variable reduction, the degree of the new polynomial is poly(sd).

Next lemma shows that the existence of an exponentially hard but E-computable polynomial family has an interesting complexity consequence. It is based on Valiant's criterion.

Lemma 10 (Valiant class separation). *If we have a polynomial family $\{f_n\}_{n \geq 1}$ that is E-computable, but has algebraic circuit complexity $2^{\Omega(n)}$, then either E$\not\subseteq$#P/poly or VNP has polynomials of algebraic circuit complexity $2^{\Omega(n)}$.*

Proof. Say, for a constant $\delta \geq 1$, we have an E-computable multi-δ-ic polynomial family $\{f_n\}_{n \geq 1}$ with algebraic circuit complexity $2^{\Omega(n)}$. Clearly, the coefficients in f_n have bitsize $2^{O(n)}$. By using a simple transformation, given in [42, Lem.3.9], we get a multilinear polynomial family $\{h_n\}_{n \geq 1}$, that is E-computable and has algebraic complexity $2^{\Omega(n)}$, such that its coefficients are $\{0, 1\}$.

Assume E$\subseteq$#P/poly. Since each coefficient of h_n is 0 or 1 that is computable in E, we deduce that the coefficient-function of h_n is in #P/poly. Thus, by [11, Prop.2.20], $\{h_n\}_{n \geq 1}$ is in VNP and has algebraic circuit complexity $2^{\Omega(n)}$. □

Next lemma converts a monomial into a sum of powers. It is called Fischer's trick in [30]. It requires char $\mathbb{F} = 0$ or large.

Lemma 11 (Fischer's trick [18]). *Over a field $\mathbb{F}$ of char$(\mathbb{F}) = 0$ or $> r$, any expression of the form $g = \sum_{i \in [k]} \prod_{j \in [r]} g_{ij}$ with $\deg(g_{ij}) \leq \delta$, can be rewritten as $g = \sum_{i \in [k']} c_i g_i^r$ where $k' := k2^r$, $\deg(g_i) \leq \delta$ and $c_i \in \mathbb{F}$. In fact, each g_i is a linear combination of $\{g_{i'j} | j\}$ for some i'.*

3 MANY-FOLD COMPOSITION OF NW DESIGN– PROOF OF THEOREM 1

Lemma 9 gave an efficient variable reduction from an exponentially hard but E-computable polynomial family. However, while bootstrapping in Theorem 1, we work with a case where number of variables can be as low as $\log^{\circ c} s$ compared to s, size of the circuit. In this extremely low variate regime, we have to deal with hard polynomial family of *non*-constant individual degree. There are also technical challenges faced in choosing parameters that should lead to a contradiction in the proof. So, we cannot use Lemma 9 directly. In Lemma 12, we take care of those issues. Overall proof strategy will be again to use Nisan-Wigderson combinatorial design and Kaltofen's algebraic circuit factoring algorithm. This is done repeatedly in Theorem 1.

Lemma 12 (Tiny variable reduction). *Let $c_3 \geq 1$ be the exponent in Kaltofen's factoring algorithm [11, Thm.2.21]. For a constant $e \geq 1$ define, $c_0 := \lceil 9\sqrt{e} + 3 \rceil c_3$, $c_1 := \lceil 30e + 10\sqrt{e+1} \rceil c_3$ and $c_2 := 1 + c_1^2$. Let ε be a tiny function say $\varepsilon(s) := 2\lceil \log^{\circ k} s \rceil$ for $k \geq 1$. Suppose we have a family $\{q_{m,s} \mid s \in \mathbb{N},\ m = c_1 \varepsilon(s)\}$ of multi-$\delta_{m,s}$-ic m-variate polynomials that can be computed in $s^{O(1)}$ time, but has no size-s algebraic circuit, where $\delta_{m,s} := \lceil s^{3e/m} \rceil$.*

Then, there is a poly(sd)-time variable reduction map, reducing $n \leq 2^{\varepsilon((sd)^{c_0})}$ to $c_2 \varepsilon((sd)^{c_0})$ and preserving nonzeroness, for degree-d n-variate polynomials computed by size-s circuits. Furthermore, after variable reduction, the degree of the new polynomial will be poly(sd).

Proof. Let $s' := sd$. Let $\mathcal{P}$ be the set of degree-d polynomials computed by size-s circuits that depend only on the first n-variables. We intend to stretch $c_2 \varepsilon(s'^{c_0})$ variables to n. Define $m' := c_1 \varepsilon((sd)^{c_0})$. Note that $q := q_{m', s'^{c_0}}$ has no algebraic circuit of size s'^{c_0}. Its individual-degree is $\leq \delta := \lceil s'^{3ec_0/m'} \rceil = s'^{o(1)}$.

Let $\mathcal{D} = \{S_1, \ldots, S_n\}$ be a $(c_2 \varepsilon(s'^{c_0}), m', 10\varepsilon(s'^{c_0}))$-design on the variable set $Z = \{z_1, \ldots, z_{c_2 \varepsilon(s'^{c_0})}\}$. Constants $c_2 > c_1 > 10$ will ensure the existence of the design by Lemma 8. Our hitting-set generator for $\mathcal{P}$ is defined as: for all $i \in [n]$, $x_i \mapsto q(S_i) =: p_i$ with S_i as variables. Then, we show that for any nonzero polynomial $P(\mathbf{x}) \in \mathcal{P}$, $P(p_1, \ldots, p_n)$ is also nonzero.

For the sake of contradiction, assume that $P(p_1,\dots,p_n)$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [n]$ such that $P(p_1,\dots,p_{j-1},x_j,\dots,x_n) =: P_1$ is nonzero, but $P_1\big|_{x_j=p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be a constant assignment on all the variables in P_1, except x_j and the variables S_j in p_j, with the property: P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, we can find such an assignment [64]. Now our new polynomial P_2 on the variables S_j and x_j is of the form $P_2(S_j,x_j) = P(p'_1,\dots,p'_{j-1},x_j,a_{j+1},\dots,a_n)$, where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_i's are field constants decided by our assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \le 10\varepsilon(s'^{c_0})$. Since p'_i are polynomials on variables $S_i \cap S_j$ of individual degree$\le \delta$, each p'_i has a circuit (of trivial form $\Sigma\Pi$) of size at most $m'\delta \cdot \delta^{10\varepsilon(s'^{c_0})} = m'\delta \cdot \delta^{10m'/c_1}$.

Thus, we have a circuit for P_2 of size at most $s_1 := s + nm'\delta \cdot \delta^{10m'/c_1}$, and degree of the computed polynomial is at most $d_1 := dm'\delta$. Since $(x_j - p_j)$ divides P_2, we can invoke Kaltofen's factorization algorithm [39] (see [11, Theorem 2.21] for the algebraic circuit complexity of factors) and get an algebraic circuit for p_j of size $(s_1d_1)^{c_3}$, which is

$$\begin{aligned} &\le (snm'\delta \cdot \delta^{10m'/c_1} \cdot dm'\delta)^{c_3} \\ &= \left(s'nm'^2\delta^{2+\frac{10m'}{c_1}}\right)^{c_3} \\ &< (s'^{2+o(1)} \cdot \delta^{10m'/c_1})^{c_3} < s'^{(3+30ec_0/c_1)c_3}. \end{aligned}$$

This exponent

$$\begin{aligned} &= \left(\frac{3}{\lceil(9\sqrt{e}+3)\rceil} + \frac{30e}{\lceil 30e + 10\sqrt{e+1}\rceil}\right) c_0 \\ &\le \left(\frac{1}{(3\sqrt{e}+1)} + \frac{3\sqrt{e}}{3\sqrt{e}+\sqrt{1+1/e}}\right) c_0 < c_0. \end{aligned}$$

So, $p_j = q(S_j)$ has circuit of size smaller than s'^{c_0}, which contradicts the hardness of q. Thus, $C(p_1,\dots,p_n)$ is nonzero.

The time for computing $(p_1,\dots,p_n)$ depends on: **(1)** computing the design (i.e. poly$(2^{m'})$-time), and **(2)** computing q (i.e. poly(sd)-time). Thus, the variable reduction map is computable in $\delta^{O(m')} =$ poly(sd)-time. After variable reduction, the degree of the new polynomial is $< nd \cdot \deg(q) =$ poly(sd). □

Remark. In the case of a finite field $\mathbb{F} = \mathbb{F}_{r^t}$ of prime characteristic r, we have to be careful while invoking Kaltofen's factoring. As, the latter outputs a small circuit for $p_j^{r^{t'}}$ where $r^{t'}$ is the highest power dividing the multiplicity of $x_j - p_j$ in P_2. However, when we raise the output by $r^{t-t'}$ we get a circuit that is small and agrees with p_j on $\mathbb{F}$-points. This is used, like in [38, Rmk.7.5], to redefine algebraic complexity of q over $\mathbb{F}_{r^t}$ suitably and the above lemma works.

Proof of Theorem 1. Consider the following two statements. **S1:** we have a poly(s)-time hsg for size-s degree-s circuits that depend only on the first $\lceil \log^{\circ c} s\rceil$ variables, and **S2:** we have a poly(s)-time hsg for degree-s polynomials computed by size-s circuits that depend only on the first $\lceil \log^{\circ c} s\rceil$ variables. **S1** is our given hypothesis. However, in this proof, we work with **S2** which is stronger than **S1**, as in the former case circuits may have degree *larger* than s. So we first argue that they are equivalent up to polynomial overhead. **S2** trivially implies **S1**. For the other direction, we invoke (the proof of) the 'log-depth reduction' result for arithmetic circuits. For any size-s circuit C computing a degree-s polynomial, we have an s^{e_0}-size s-degree circuit C' computing the same polynomial, for some constant e_0 (see [59, Thm.5.15]). Now apply **S1** for s^{e_0}-size s-degree and get poly(s)-hsg for C. Next, we focus on designing poly(sd)-hsg for degree-d polynomials computed by size-s circuits, using our stronger hypothesis **S2**.

Suppose that for all $s,d,i \in \mathbb{N}$, $\mathcal{P}_{s,d,i}$ is the set of degree-d polynomials computed by size-s circuits that depend only on the first $f_i(sd)$ variables, where $f_i(s) := (\log^{\circ i} s)^2$. We prove that for all $0 \le i \le c+1$, we have a polynomial time hitting set for $\mathcal{P}_{s,d,i}$. We will use reverse induction on i. Define function $\varepsilon_i(s) := 2\lceil \log^{\circ i} s\rceil$.

Base case– Poly(sd)-hsg for $\mathcal{P}_{s,d,c+1}$: Let $t := \max\{s,d\}$. Then the set of polynomials $\mathcal{P}_{s,d,c+1}$ is a subset of $\mathcal{P}_{t,t,c+1}$. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of degree-s polynomials computed by s-size circuits that depend only on the first $\lceil \log^{\circ c} s\rceil$ variables. Using the hypothesis **S2**, we have a poly(s) time hsg for $\mathcal{T}_s$. Since $f_{c+1}(t) \le \lceil \log^{\circ c} t\rceil$ for large t, $\mathcal{P}_{t,t,c+1}$ is a subset of $\mathcal{T}_t$. So $\mathcal{P}_{t,t,c+1}$ also has a poly(t)-time hsg. This gives a poly(sd)-time hsg for $\mathcal{P}_{s,d,c+1}$.

Induction step– From poly$(s'd')$-hsg of $\mathcal{P}_{s',d',i}$ to poly(sd)-hsg of $\mathcal{P}_{s,d,i-1}$: We divide this step into three smaller steps, for $i \in [c+1]$.

1) Hsg of $\mathcal{P}_{s',d',i}$ to hard polynomial family: For some constant e, we have $((s'd')^{e/2},(s'd')^{e/2})$-hsg for $\mathcal{P}_{s',d',i}$. Let for all $s,i \in \mathbb{N}$, $\mathcal{T}_{s,i}$ be the set of degree-s polynomials computed by size-s circuits that depend only on the first $c_1\varepsilon_i(s)$ variables, where c_1 is a constant as defined in Lemma 12 using the e. Note that $m := c_1\varepsilon_i(s)$ is smaller than $f_i(s^2)$ for large enough s. So, polynomial time hsg for $\mathcal{P}_{s',d',i}$ gives a (s^e,s^e)-hsg for $\mathcal{T}_{s,i}$. Then using Lemma 5, we get an m-variate polynomial $q_{m,s}$ such that **1)** individual degree is less than $\delta_{m,s} = \lceil s^{3e/m}\rceil$, **2)** $q_{m,s} \notin \mathcal{T}_{s,i}$, and **3)** computable in $s^{O(1)}$-time.

Suppose $q_{m,s}$ has a circuit C of size less than s. Since the degree $(m \cdot \delta_{m,s})$ is also less than s, the polynomial $q_{m,s}$ is in $\mathcal{T}_{s,i}$, which is a contradiction. So using (s^e,s^e)-hsg for $\mathcal{T}_{s,i}$, for all $s \in \mathbb{N}$, we get a polynomial family $\{q_{m,s} \mid s \in \mathbb{N},\ m = c_1\varepsilon_i(s)\}$ of multi-$\delta_{m,s}$-ic that can be computed in $s^{O(1)}$ time, but has no size-s algebraic circuit.

2) Hard polynomial to variable reduction map: Note that $f_{i-1}(sd) \le 2^{\varepsilon_i((sd)^{c_0})}$, where c_0 is a constant defined in Lemma 12 using the e. Using the lemma (for $\varepsilon = \varepsilon_i$), any nonzero polynomial $P \in \mathcal{P}_{s,d,i-1}$ can be converted, in poly(sd)-time, to another poly(sd)-degree nonzero polynomial P' computed by poly(sd)-size circuit which depends only on the first $c_2\varepsilon_i((sd)^{c_0})$ variables.

3) The map to poly(sd) time hsg for $\mathcal{P}_{s,d,i-1}$: Since, in P', the number of variables $c_2\varepsilon_i((sd)^{c_0})$ is less than $f_i(sd)$, using poly-time hsg of $\mathcal{P}_{s',d',i}$ we get a poly-time hsg for $P \in \mathcal{P}_{s,d,i-1}$.

Repetition– After applying the induction step $c+1$ times, we have a poly(sd)-time hsg for $\mathcal{P}_{s,d,0}$. In other words, we have a poly(sd)-time hsg for size-s degree-d circuits.

Now we show that Conjecture 1 holds. We just obtained a poly(s)-time hsg for $\mathcal{T}_{s,1}$. Let $m = \lceil \log s\rceil$. Then applying Lemma 5, we get a family of polynomials $\{q_m\}_{m\ge1}$ such that **1)** it is multi-δ-ic, for some constant δ, and **2)** computable in $\delta^{O(m)}$-time, but has no $\delta^{o(m)}$-size algebraic circuit. Now, applying Lemma 10, we get Conjecture 1. □

Remark. In the case of a finite field $\mathbb{F} = \mathbb{F}_{r^t}$ of prime characteristic r, we have to redefine the hardness of the polynomial $q_{m,s}$ in Step (1) of the induction step above. As remarked before, we can define $\mathcal{T}_{s,i}$ to be the set of polynomials $f(x_1, \ldots, x_{c_1 \varepsilon_i(s)})$, such that for some e, f^{r^e} agrees *on all* $\mathbb{F}$*-points* with some nonzero degree-s polynomial computed by a size-s circuit. It can be seen that an hsg for $\mathcal{T}_{s,i}$ gives a hard $q_{m,s}$ (via the annihilator approach of Lemma 5) that can be used in Step (2).

Next, we relax the hypothesis of Theorem 1 by allowing a subexponential time hitting-set. In the following discussion, we use a constant e_0. It is the exponent of 'log-depth reduction' algorithm ([59, Thm.5.15]), i.e. for every size-s circuit computing a degree-d n-variate polynomial, we also have an $(sdn)^{e_0}$-size d-degree circuit computing the same polynomial. We recall the following standard definition.

Subexp. A function $f(s)$ is in *subexp* if $f(s) = \exp(s^{o(1)})$. Eg. $2^{\sqrt{s}} \notin$ subexp, but $\exp(2^{\sqrt{\log s}}) \in$ subexp. One can recall the standard complexity class, SUBEXP $:= \cap_{\epsilon>0}$DTIME$(\exp(n^{\epsilon}))$. Basically, these are decision problems whose time-complexity is a subexp function.

Theorem 13 (Subexp bootstrap). *Let f be a function in subexp. Suppose that we have a poly$(f(s))$ time blackbox PIT for size-s degree-s circuits that depend only on the first $10\lceil \log f(s) \rceil$ variables. Then, we have blackbox PIT for size-s degree-d circuits in subexponential, $\exp((sd)^{o(1)})$, time. Furthermore, either $E \not\subseteq$ #P/poly or VNP≠VP.*

Remark. As an fpt-algorithm the hypothesis requires a blackbox PIT, for size-s degree-s n-variate circuits, of time complexity potentially as large as $\exp(s^{o(1)} + O(n))$.

For proof, see the full version linked on the first page.

4 BOOTSTRAP CONSTANT-VARIATE PIT– PROOF OF THEOREMS 2 & 3

The overall strategy is similar to the last section. However, the details would now change drastically.

First, we describe an optimized version of the Nisan-Wigderson (NW) design, where the parameters are different from that in Lemma 8. Later, it will help us improve the constants.

Lemma 14 (NW design). *There exists an algorithm which takes (ℓ, n, d), with $\ell \geq 100$ and $d \geq 13$, and a base set S of size $\ell := \lceil 4n^2/d \rceil$ as input, and outputs an (ℓ, n, d)-design $\mathcal{D}$ having $\geq 2^{d/4}$ subsets, in time $O((4\ell/n)^n)$.*

For the proof, see the full version linked on the first page. **Exponent vs Variables.** In this section, to describe the complexity parameters of the circuits and the hsg, we use two families of functions $\{f_i\}_{i\geq 0}$ ("exponent of time") and $\{m_i\}_{i\geq 0}$ ("number of variables"). They are defined as follows: $f_0 \geq 2$ and $m_0 \geq 1024$ are constants and for all $i \geq 1$,

$$f_i := 16f_{i-1}^2 \text{ and } m_i := 2^{m_{i-1}/(64f_{i-1}^2)}.$$

Our strategy is to use an NW $(m_i, \frac{m_i}{8f_i}, \frac{m_i}{16f_i^2})$-design to stretch m_i variables to m_{i+1}. We want to show that m_i grows much faster in contrast to f_i. In particular, we need m_i to be a *tetration* in i (i.e. iterated exponentiation), while f_i is "merely" a double-exponentiation in i. Seeing this needs some effort and we will do this in the next two propositions.

From now on we will assume that ϵ is a constant fraction satisfying $1 > \epsilon \geq (3 + 6\log(128f_i^2))/(128f_i^2)$, for $i = 0$. Since f_i increases with i, the fraction $(3 + 6\log(128f_i^2))/(128f_i^2)$ decreases. Thus, the constant ϵ remains larger than the latter, for all $i \geq 0$. Now, we describe some properties about these two sequences of numbers. For proofs, see the full version linked on the first page.

Proposition 15. *If, for some $i \geq 0$, $m_i \geq 192f_i^2 \cdot \frac{1}{\epsilon}\log(128f_i^2)$, then the same relation holds between m_{i+1} and f_{i+1}.*

Proposition 16 (m_i is a tetration). *Let $m_0 \geq \max\{(8f_0)^{\frac{2}{\epsilon}}, 192f_0^2 \cdot \frac{1}{\epsilon}\log(128f_0^2)\}$. Then for all $i \geq 0$: 1) $m_{i+1} \geq 2^{m_i^{1-\epsilon}}$ and 2) $m_{i+1} \geq 2m_i > 3456f_i^2$.*

Once we know that m_i grows extremely rapidly, we want to estimate the number of iterations before which it reaches s.

Proposition 17 (Iteration count). *The least i, for which $m_i \geq s$, is $\leq \frac{3}{1-\epsilon}\log\left(\frac{3}{1-\epsilon}\right) + 2\log^{\star} s$.*

Now we describe the i-th step of bootstrapping.

Lemma 18 (Induction step). *Let s be the input size parameter, $i \geq 0$, $m_i = s^{o(1)}$ and $m' := \min\{m_{i+1}, s\}$. Suppose that we have an s^{f_i}-time hsg for m_i-variate degree-s polynomials computed by size-s circuits. Then, we have an $s^{f_{i+1}}$-time hsg for m'-variate degree-s polynomials computed by size-s circuits.*

Proof. Although i might grow (extremely) slowly wrt s, it helps to think of i and s as two independent parameters. Suppose that for all $s \in \mathbb{N}$, $\mathcal{P}_{s,i}$ is the set of m_i-variate degree-s polynomials computed by size-s circuits, and $\mathcal{P}_{s,i+1}$ is the set of m'-variate degree-s polynomials computed by size-s circuits. Our proof can be broken into three main *steps*. First, using the hsg of $\mathcal{P}_{s,i}$ we construct a hard polynomial family. Next, using that hard polynomial family we do variable reduction on the polynomials in $\mathcal{P}_{s,i+1}$. This variable reduction is relatively "low"-cost and it reduces a *nonzero* polynomial in $\mathcal{P}_{s,i+1}$ to some *nonzero* polynomial in $\mathcal{P}_{s^{9f_i},i}$, for sufficiently large value of s. Finally, we apply the hsg of $\mathcal{P}_{s^{9f_i},i}$ to get the desired hsg for $\mathcal{P}_{s,i+1}$. The challenge is to analyze this; which we do now in detail. Keep in mind the properties of the functions m_i, f_i that we proved before.

Hard Polynomial Family Construction. We describe the construction of a hard polynomial family from the hsg of $\mathcal{P}_{s,i}$. Let $d_i(s) := s^{f_i}$ and for all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of $\frac{m_i}{8f_i}$-variate degree-s polynomials computed by size-s circuits. The $d_i(s)$-time hsg of $\mathcal{P}_{s,i}$ also gives an hsg for $\mathcal{T}_s$ with same time complexity. Like Lemma 5, the annihilator of the hsg of $\mathcal{T}_s$ gives a polynomial q_s such that: 1) $q_s \notin \mathcal{T}_s$, 2) it is computable in $d_i^4(s)$-time by linear algebra, and 3) it is multi-δ_s-ic, where $\delta_s := 1 + d_i(s)^{\frac{8f_i+1}{m_i}} = 1 + s^{f_i(8f_i+1)/m_i}$. Here, the main difference is that the individual degree bound δ_s is smaller than what Lemma 5 ensures. It will help us reduce the initial constants in our calculations. We give a brief sketch of how we get an annihilator with this individual degree.

The number of monomials on $\frac{m_i}{8f_i}$ variables with individual degree $< \delta_s$ is at least $m := d_i^{1+\frac{1}{8f_i}} = s^{f_i+\frac{1}{8}}$. After evaluating an $\frac{m_i}{8f_i}$-variate multi-δ_s-ic polynomial on the hsg of $\mathcal{T}_s$, we get a univariate polynomial of degree at most $d := \frac{m_i}{8f_i} \cdot \left(d_i + d_i^{1+\frac{8f_i+1}{m_i}}\right) \leq \frac{m_i}{8f_i} \cdot 2s^{f_i+\frac{8f_i^2+f_i}{m_i}}$. To make the linear algebra argument of Lemma 5 work, we need $m > d$. This holds as $m_i = s^{o(1)}$ and as by Proposition 16 we have $m_i \geq 1728f_i^2$.

Now we argue that q_s has no circuit of size$\leq s$. For the sake of contradiction, assume that q_s has a circuit of size$\leq s$. The degree of q_s is at most $\frac{m_i}{8f_i} \cdot 2d_i^{\frac{8f_i+1}{m_i}} \leq \frac{m_i}{8f_i} \cdot 2s^{\frac{f_i(8f_i+1)}{m_i}}$. Applying $m_i = s^{o(1)}$ and $m_i \geq 1728f_i^2$, we get that q_s has degree $< s$. This implies that $q_s \in \mathcal{T}_s$, which is a contradiction. Thus, q_s has no circuit of size$\leq s$. So we have a multi-δ_s-ic polynomial family $\{q_s \mid s \in \mathbb{N}\}$ such that, 1) q_s is computable in $d_i^4 = s^{4f_i}$ time but has no circuit of size$\leq s$, 2) it has individual degree $\delta_s = 1 + s^{f_i(8f_i+1)/m_i}$ and number of variables $\frac{m_i}{8f_i}$.

Variable Reduction Map. Now we convert every non-zero polynomial in $\mathcal{P}_{s,i+1}$ to a non-zero polynomial in $\mathcal{P}_{s^{12f_i},i}$. Consider a slightly larger size parameter $s_0 := s^7$. Let $\{S_1,\ldots,S_{m'}\}$ be an NW $(m_i, \frac{m_i}{8f_i}, \frac{m_i}{16f_i^2})$-design on the variable set $\{z_1,\ldots,z_{m_i}\}$. The growth properties of m_i, togetherwith Lemma 14, ensures that such a design exists. Define for all $j \in [m'], p_j := q_{s_0}(S_j)$. Next, we show that for any non-zero $P \in \mathcal{P}_{s,i+1}$, $P(p_1,\ldots,p_{m'})$ is also non-zero.

For the sake of contradiction, assume that $P(p_1,\ldots,p_{m'})$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [m']$ such that $P(p_1,\ldots,p_{j-1},x_j,\ldots,x_{m'}) =: P_1$ is nonzero, but $P_1\big|_{x_j=p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be a constant assignment on all the variables in P_1, except x_j and the variables S_j in p_j, with the property: P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, we can find such an assignment [64]. Now our new polynomial P_2, on the variables S_j and x_j, is of the form $P_2(S_j,x_j) := P(p'_1,\ldots,p'_{j-1},x_j,a_{j+1},\ldots,a_{m'})$, where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and a_i's are field constants decided by our assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \leq \frac{m_i}{16f_i^2}$. Since p_is are polynomials on variables S_i of individual degree$\leq \delta_{s_0}$, each p'_i has a circuit (of trivial form $\Sigma\Pi$) of size at most

$$\frac{m_i}{16f_i^2}\delta_{s_0} \cdot \delta_{s_0}^{\frac{m_i}{16f_i^2}} .$$

Thus, we have a circuit for P_2 of size at most s_1 and the degree of P_2 is at most d_1, where

$$s_1 := s + \frac{m' m_i \delta_{s_0}}{16f_i^2} \cdot \delta_{s_0}^{\frac{m_i}{16f_i^2}} \quad \text{and} \quad d_1 := s \cdot \frac{m_i \delta_{s_0}}{16f_i^2} .$$

Since $(x_j - p_j)$ divides P_2, we can invoke Kaltofen's factorization algorithm [39] (see [11, Thm.2.21] for the improved complexity of factors) and get an algebraic circuit for p_j of size $s'_0 := s_1\tilde{O}(d_1^2)$. Now we prove that $s'_0 < s_0$, for large enough s. This implies that q_{s_0} has a circuit of size$\leq s_0$ which contradicts the hardness of q_{s_0}.

Recall that $\delta_{s_0} = 1 + s_0^{f_i(8f_i+1)/m_i}$. Let us upper bound $s'_0 =$

$$\begin{aligned}
s_1\tilde{O}(d_1^2) &\leq \left(s + \frac{m' m_i}{16f_i^2} \cdot \delta_{s_0}^{1+\frac{m_i}{16f_i^2}}\right) \cdot \tilde{O}\left(\frac{s m_i \delta_{s_0}}{16f_i^2}\right)^2 \\
&\leq \frac{s^{3+o(1)}\delta_{s_0}^2}{f_i^2} + \frac{s^{3+o(1)}\delta_{s_0}^{3+\frac{m_i}{16f_i^2}}}{f_i^4} \quad \left(\because m_i = s^{o(1)}, m' \leq s\right) \\
&\leq s^{3+o(1)} s_0^{\frac{2f_i(8f_i+1)}{m_i}} + s^{3+o(1)} s_0^{\left(3+\frac{m_i}{16f_i^2}\right)\frac{f_i(8f_i+1)}{m_i}} \\
&\leq s^{3+o(1)} s^{\frac{14f_i(8f_i+1)}{m_i}} + s^{3+o(1)} s^{\left(3+\frac{m_i}{16f_i^2}\right)\frac{7f_i(8f_i+1)}{m_i}} \\
&\leq s^{3+o(1)} s^{\frac{112f_i^2+14f_i}{m_i}} + s^{3+o(1)} s^{\frac{21f_i(8f_i+1)}{m_i}+\frac{7(8f_i+1)}{16f_i}}
\end{aligned}$$

$$\begin{aligned}
&\leq s^{3+o(1)} s^{\frac{112f_i+14}{1728f_i}} + s^{3+o(1)} s^{\frac{21(8f_i+1)}{1728f_i}+\frac{7(8f_i+1)}{16f_i}} \quad \left(\because m_i > 1728f_i^2\right) \\
&\leq s^{3+o(1)+\frac{112}{1728}+\frac{7}{1728}} + s^{3+o(1)+\frac{168}{1728}+\frac{21}{3456}+\frac{56}{16}+\frac{7}{32}} \quad (\because f_i \geq 2) \\
&\leq s^{3.1+o(1)} + s^{6.83+o(1)} \\
&< s^7 = s_0 .
\end{aligned}$$

This gives a contradiction for sufficiently large s. So $P' := P(p_1, \ldots,p_{m'})$ is non-zero.

Using the Given Hsg. The above variable reduction converts P to a m_i-variate degree-d' non-zero polynomial P' computable by s'-size circuit, where

$$d' := \frac{s m_i}{8f_i} \cdot \delta_{s_0} \quad \text{and} \quad s' := s + \frac{m' m_i \delta_{s_0}}{8f_i} \cdot \delta_{s_0}^{\frac{m_i}{8f_i}} .$$

Now we give an upper bound of s':

$$\begin{aligned}
s' &= s + \frac{m' m_i \delta_{s_0}}{8f_i} \cdot \delta_{s_0}^{\frac{m_i}{8f_i}} \\
&= s + \frac{m' m_i}{8f_i} \cdot \left(1 + s_0^{\frac{f_i(8f_i+1)}{m_i}}\right)^{(\frac{m_i}{8f_i}+1)} \\
&\leq s + \frac{m' m_i}{8f_i} \cdot (1+s)^{\frac{7f_i(8f_i+1)}{m_i}(\frac{m_i}{8f_i}+1)} \\
&\leq s + s^{1+o(1)+7(f_i+\frac{1}{8})(1+\frac{8f_i}{m_i})} \\
&< s + s^{1+o(1)+7(f_i+\frac{1}{8})(1+\frac{8}{3456})} \quad (\because m_i > 1728f_i^2, f_i \geq 2) \\
&< s^{9f_i} .
\end{aligned}$$

Since $d',s' < s^{9f_i} =: s_1$, P' is m_i-variate degree-s_1 polynomial that is computable by size-s_1 circuit. So P' has an hsg of time complexity $s_1^{f_i} = s^{9f_i^2}$.

Final Time Complexity. First, let us review our overall algorithm: It takes $(1^s, 1^{i+1})$ as input, and in $s^{f_{i+1}}$-time, outputs an $s^{9f_i^2}$-time hsg of $\mathcal{P}_{s,i+1}$, under the assumption that for all $t \geq s$, there is a t^{f_i}-time hsg for $\mathcal{P}_{t,i}$.

a. $s_0 \leftarrow s^7$.
b. By linear algebra, compute an annihilator q_{s_0} (in dense representation) of the given hsg of $\frac{m_i}{8f_i}$-variate degree-s_0 size-s_0 polynomials.

c. Compute NW design (by the greedy algorithm sketched in Lemma 14) $\{S_1, \ldots, S_{m'}\}$ on the variable set $\{z_1, \ldots, z_{m_i}\}$.
d. Compute an m_i-input and m'-output circuit C (in the form $\Sigma\Pi$) on the variables $\{z_1, \ldots, z_{m_i}\}$ such that: for all $j \in [m']$, the j-th output is $p_j = q_{s_0}(S_j)$.
e. Compute the hsg $\mathbf{a} = (a_1, \ldots, a_{m_i})$ of $\mathcal{P}_{s^{9f_i},i}$. Then, the above proof shows that an hsg for $\mathcal{P}_{s,i+1}$ is $C(\mathbf{a})$.

The total time complexity of hsg for P has four components:

(1) Computing q_{s_0} (step b): It takes $(s_0^{f_i})^4 = s^{7\times f_i \times 4} = s^{28 f_i} \le s^{14 f_i^2}$.
(2) Nisan Wigderson design from Lemma 14 (step c): It takes time $O\left(4m_i/(m_i/8f_i)\right)^{m_i/8f_i} = O(32 f_i)^{m_i/8f_i}$. If $m_i > 64 f_i^2 \log s$ then we will run the i-th induction step only for (relabelled) $m_i := 64 f_i^2 \log s$, as the stretch obtained will already be to $2^{m_i/64 f_i^2} = s$ variables. Note that at that point, i would be non-constant and hence $f_i > 4$. In this regime, $(32 f_i)^{m_i/8f_i} = (32 f_i)^{8 f_i \log s} = s^{8 f_i \log(32 f_i)} < s^{12 f_i^2}$.
(3) Computing C (step d): Essentially, compute m' copies of q_{s_0} (in dense representation). As seen before, the total time-complexity is $s^{9 f_i}$.
(4) Computing hsg of $\mathcal{P}_{s^{9f_i},i}$. Then, computing hsg of $\mathcal{P}_{s,i+1}$ by composition (step e): It takes $s^{9 f_i^2} + s^{9 f_i^2} \cdot s^{9 f_i} < s^{14 f_i^2}$ time.

So, the total time is smaller than $s^{16 f_i^2} = s^{f_{i+1}}$ and we have an hsg for m'-variate P. □

Proof of Theorem 2. In the hypothesis of the theorem statement we are given constants $e \ge 2$ and $n \ge 1024$. Let us define the m_i, f_i polynomial family with the initialization $f_0 := e$ and $m_0 := n$. The idea is to simply use the induction step (Lemma 18) several times to boost m_0 variables to an arbitrary amount.

Let P be a degree-s polynomial computed by size-s circuit. Then, it can have at most s variables. Let k be the smallest integer such that $m_k \ge s$ (k is an extremely slow growing function in s as described in Proposition 17). By Proposition 16, we have that $m_{k-1} \le s^{o(1)}$.

For $i \in \mathbb{N}_{\ge 0}$ and large enough parameters $t > t' > s$, let $\mathcal{P}_{t,i}$ denote the set of m_i-variate degree-t polynomials computed by size-t circuits. From the hypothesis, we have a t^{f_0}-time hsg for $\mathcal{P}_{t,0}$. Now for each $i < k$, we apply Lemma 18, to get the $t'^{f_{i+1}}$-hsg for $\mathcal{P}_{t',i+1}$. After k such applications of Lemma 18, we get an s^{f_k}-time hsg for *s-variate* degree-s polynomials computed by size-s circuits.

Note that $f_k = (16 f_0)^{2^k}/16 = 2^{O(2^k)} = 2^{2^{O(\log^\star s)}}$. Thus, we have an $s^{\exp\circ\exp(O(\log^\star s))}$-time blackbox PIT for VP circuits.

Since $f_0 < m_0/2$ one can see that the hypothesis of Theorem 4 is easily satisfied. This gives us an E-computable polynomial family $\{q_m\}_{m\ge1}$ with hardness $2^{\Omega(m)}$. □

Proof of Theorem 3. Suppose we have, for constant $\delta < 1/2$, an s^{n^δ}-time hsg for size-s degree-s circuits that depend only on the first n variables. Wlog (using depth-reduction proofs), we can assume that we have an s^{n^δ}-time hsg for degree-s polynomials computed by size-s circuits that depend only on the first n variables.

Then, there exists an $\epsilon \in [2\delta, 1)$ and a large enough *constant e* such that: there is an s^e-time hsg for degree-s polynomials computed by size-s circuits that depend only on the first $n := \lceil (64e^2)^{1/\epsilon} \rceil \ge 192 e^2 \log(128 e^2)^{1/\epsilon}$ variables. Note that $e \ge (n-1)^{\epsilon/2}/8 > n^\delta$ can be easily ensured, thus, s^e-time is more than s^{n^δ}-time. Now we simply invoke Theorem 2. □

Remark. 1) The NW (ℓ, n, d)-design that we are using, in the i-th iteration (Lemma 18), has its respective parameters in the "ratio" $f_i^2 : f_i : 1$ (roughly). This seems to be the reason why we need second-exponent δ slightly less than 1/2. We leave it as an open question to improve this.

2) We can give a more refined analysis in the above proofs by "decoupling" the time-complexity from the degree of the hsg. For example, we can begin with a much weaker hypothesis— for constant $\delta < 1/2$ and an arbitrarily large function $\mu(\cdot)$, an $\left(s^{\mu(n)}, s^{n^\delta}\right)$-hsg for size-$s$ degree-s circuits that depend only on the first n variables —and still get the same conclusion as in Theorem 3. This will require analysing the bit complexity (i.e. time) and the algebraic complexity (i.e. degree of the hsg) separately in the proof of Lemma 18. We skip the details for now.

5 SHALLOW DEPTHS, TINY VARIABLES– PROOF OF THEOREM 4

Shallow circuits. *Diagonal depth*-4 circuits compute polynomials of the form $\sum_{i\in[k]} c_i f_i^{a_i}$ where f_i's are sparse polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $\le b$, $a_i \le a$ and c_i's in $\mathbb{F}$. A standard notation to denote this class is $\Sigma\wedge^a\Sigma\Pi^b(n)$. This is a special case of the *depth*-4 $\Sigma^k\Pi^a\Sigma\Pi^b(n)$ model that computes polynomials of the form $\sum_{i\in[k]}\prod_{j\in[a]} f_{i,j}$ where $f_{i,j}$'s are sparse polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $\le b$. The superscripts k, a, b on the gates denote an upper bound on the respective fanin (whenever it needs to be emphasized).

We denote $\Sigma\Pi\Sigma\Pi^1$ circuits by $\Sigma\Pi\Sigma$ and call them *depth*-3. We also study a model quite close to it– $\Sigma\Pi\Sigma\wedge^b$ –we call it *preprocessed depth*-3 because, in this work, this model will appear on simply substituting univariate monomials in the variables of a depth-3 circuit. It degenerates to depth-3 again if $b = 1$.

We prove Theorem 4 in two different ways. First, by assuming an efficient $O(s^{n/2}/\log^2 s)$-degree hsg for polynomials computed by size-s $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first n variables ($a(s)$ is an arbitrarily slow growing function), we get to the conclusion of Theorem 4. Second, by assuming an efficient $O(s^{n/2}/\log^2 s)$-degree hsg for polynomials computed by size-s $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables, we get to the conclusion of Theorem 4. Both the models seem weaker than general depth-4 circuits. So one would expect that solving PIT for these models would be easier.

Our proofs will go via a plethora of intermediate models. Theorems 19 & 20 together give the proof of our first approach. Theorems 21 & 22 together give the proof of the second approach. One can notice that in all these theorems we prove the existence of an efficient variable reduction map for circuits that preserves nonzeroness. It is stronger than proving quasipolynomial hsg for size-s degree-d circuits. However, after the variable reduction, if we apply hsg of the trivial PIT derandomization [64], we get an $(sd)^{O(\log(sd))}$ time hsg.

Theorem 19 ($\Sigma\wedge^a\Sigma\Pi$ computing multilinear). *Suppose that for some constant $n \ge 2$ and some arbitrarily slow growing function*

$a(s)$, we have a $\big(poly(s), O(s^n/\log^2 s)\big)$-hsg for multilinear *polynomials computed by size-s $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first $n\log s$ variables.*

Then, for N-variate d-degree size-s circuits, we have a poly(sd)-time nonzeroness preserving variable reducing polynomial map ($N\mapsto O(\log(sd))$) and Conjecture 1 holds. Furthermore, after variable reduction, the degree of the new polynomial will be poly(sd).

Proof sketch. The proof is along the lines of [7, Thm.3.2] and is available in the full version linked on the first page.

For all $s\in\mathbb{N}$, let $\mathcal{P}_s$ be the set of multilinear polynomials computed by size-s $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first $n\log s$ variables. First, using the $O(s^n/\log^2 s)$-degree hsg we can construct a family of multilinear polynomials $\{q_m\}_m$ which is E-computable (Lemma 5) but not computable by $2^{o(m)}$-size circuits (by 'depth-4 chasm').

Using this hard polynomial family we get both the variable reduction and Conclusion 1. Invoking Lemma 9, in poly(sd) time, we can convert a nonzero d-degree N-variate polynomial computed by a size-s circuit to a nonzero $O(\log(sd))$-variate poly(sd)-degree polynomial. Conjecture 1 follows from Lemma 10. □

Remark. 1) Note that a $\big(\tilde{O}(s^n), s^n\big)$-hsg for *multilinear* $n\log s$ variate polynomials is trivial. As one can simply use $\{0,1\}^{n\log s}$ as the hitting-set.

2) An efficient $s^n/\omega(\log s)$ degree hsg in the hypothesis would also suffice.

3) Can we get a conclusion as strong as in Theorem 1? In the proof above we get a variable reduction map to log-variate; but this map when applied on a general circuit results in a *non*-multilinear polynomial. So, we cannot use the hsg provided in the hypothesis and have to do poly(s)-time PIT on the log-variate $\Sigma\wedge^a\Sigma\Pi$ circuit by some other means (currently unknown).

Theorem 20 (Tiny variate $\Sigma\wedge^a\Sigma\Pi$). *Suppose that for some constant $n\geq 3$ and some arbitrarily slow growing function a, we have a $\big(poly(s), O(s^{n/2}/\log^2 s)\big)$-hsg for size-$s$ $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first n variables. Then, we get all the conclusions of Theorem 19.*

Proof. For all $s\in\mathbb{N}$, let $\mathcal{P}_s$ be the set of multilinear polynomials computed by size-s $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first $n\log s$ variables. For all $s\in\mathbb{N}$, let $\mathcal{T}_s$ be the set of polynomials computed by size-s $\Sigma\wedge^a\Sigma\Pi$ circuits that depend only on the first n variables. By the hypothesis, we have an efficient $O(s^{n/2}/\log^2 s)$-degree hsg for $\mathcal{T}_s$. Next, we convert every nonzero polynomial in $\mathcal{P}_s$ to a nonzero polynomial in $\mathcal{T}_{O(s^2)}$ in poly(s) time. Now applying the given hsg for $\mathcal{T}_{O(s^2)}$, we get an efficient $O(s^n/\log^2 s)$-degree hsg for $\mathcal{P}_s$. Next invoking Theorem 19, we get our conclusion.

We describe the reduction from $\mathcal{P}_s$ to $\mathcal{T}_{O(s^2)}$. Let P be a nonzero polynomial in $\mathcal{P}_s$. Let $m := n\log s$. Partition the variable set $\{x_1,\ldots,x_m\}$ into n blocks $B_j, j\in[n]$, each of size $\log s$. Let $B_j := \{x_{u(j)+1}, x_{u(j)+2},\ldots,x_{u(j)+\log s}\}$, for all $j\in[n]$ and $u(j):=(j-1)\log s$. Consider the variable-reducing "local Kronecker" map $\varphi: x_{u(j)+i}\mapsto y_j^{2^i}$. Note that $\varphi(P)\in\mathbb{F}[y_1,\ldots,y_n]$, and its individual-degree is at most $2s$. It is easy to see that $\varphi(P)\neq 0$ (basically, use the fact that P computes a nonzero multilinear polynomial and φ keeps the multilinear monomials distinct). Finally, $\varphi(P)$ becomes an n-variate $\Sigma\wedge^a\Sigma\Pi$ circuit of size at most $s+s\cdot 2^{\log s}=O(s^2)$. Thus, $\big(\text{poly}(s), O(s^n/\log^2 s)\big)$-hsg for $\mathcal{T}_{O(s^2)}$ gives a $\big(\text{poly}(s), O(\frac{s^n}{\log^2 s})\big)$-hsg for P. □

In next two lemmas, we describe our second approach to prove Theorem 4.

Theorem 21 (Depth-3 computing multilinear). *Suppose that for some constant $n\geq 2$, we have a $\big(poly(s), O(s^n/\log^2 s)\big)$-hsg for* multilinear *polynomials computed by size-s depth-3 circuits that depend only on the first $n\log s$ variables. Then, we get all the conclusions of Theorem 19.*

Proof. Proof will be similar to proof of Theorem 19. Main difference is that there we were dealing with depth-4 circuits, but here we have depth-3 circuits. So we need 'depth-3-reduction' result [30] with 'depth-4-reduction' result [7]. We only sketch the main points here.

First, we construct a hard polynomial family from the hsg. According to the hypothesis, for $n\log s$-variate multilinear polynomials computed by size-s depth-3 circuits we have an $O(s^n/\log^2 s)$-degree hsg. For all $s\in\mathbb{N}$, let $\mathcal{P}_s$ be the set of $n\log s$-variate multilinear polynomials computed by size-s depth-3 circuits. Let $m := n\log s$. Let $\mathbf{f}(y)$ be the $\big(\text{poly}(s), O(s^n/\log^2 s)\big)$-hsg of $\mathcal{P}_s$. Now we consider the annihilator of $\mathbf{f}(y)$ to get a hard polynomial. Let k be the number of m-variate $m/2$-degree multilinear monomials. Then $k=\binom{m}{m/2}\geq 2^m/\sqrt{2m}=s^n/\sqrt{2m}>O(s^n/\log^2 s)\cdot m$ (for large enough s). Thus, by linear algebra similar to Lemma 5, we get an m-variate $m/2$-degree multilinear homogeneous annihilating polynomial $q_m\notin\mathcal{P}_s$ and computable in poly(s)-time. Importantly $q_m\notin\mathcal{P}_s$, thus, no depth-3 circuit of size $<s=2^{\Theta(m)}$ can compute it. Next we show that it is also not computable by any $2^{o(m)}$-size algebraic circuit.

For the sake of contradiction, assume that q_m has a $2^{o(m)}$-size circuit. Repeat the depth-reduction arguments, as in the proof of Theorem 19, after cutting at some depth $t=\omega(1)$. Let $a:=5^t$ and $b:=m/2^{t+1}$. Here, we can also ensure $a,b=o(m)=o(\log s)$, $a=\omega(1)$, and we have a $2^{o(m)}$-size shallow circuit for q_m of the form $\Sigma\Pi^a\Sigma\Pi^b$.

It was shown in [30] that any size-s' n-variate $\Sigma\Pi^a\Sigma\Pi^b$ circuit can be transformed to a poly($s'2^{a+b}$)-size n-variate $\Sigma\Pi\Sigma^b$ circuit. Applying it here, we get a depth-3 circuit C', computing q_m, of the form $\Sigma\Pi\Sigma$ and size $2^{o(m)}\cdot 2^{a+b}=2^{o(m)}$. This gives a contradiction, since no depth-3 circuit of size $<s=2^{\Theta(m)}$ can compute it.

Thus, we have an E-computable family of multilinear polynomials $\{q_m\}_{m\geq 1}$ that has no circuit of size $2^{o(m)}$. Using this hard polynomial family we get both the variable reduction and Conjecture 1 as before. □

Theorem 22 (Tiny variate $\Sigma\Pi\Sigma\wedge$). *Suppose that for some constant $n\geq 3$, we have a $\big(poly(s), O(s^{n/2}/\log^2 s)\big)$-hsg for polynomials computed by size-s $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables. Then, we get all the conclusions of Theorem 19.*

Proof. The proof is similar to that of Theorem 20 and available in the full version linked on the first page for a proof. □

Remark. Can we get a result like the above with depth-3 circuits in the hypothesis? At this point it is not clear how to get to arbitrarily tiny variate $\Sigma\Pi\Sigma$ because: **1)** the above trick of applying local-Kronecker map, to reduce variables from $n \log s$ to n, increases the circuit depth to 4. Moreover any such map has to be *non*-linear, otherwise the resulting monomials are too few, and **2)** in the tiny variate regime we need degree $\geq \Omega(s)$ so that the hsg of the model can be used to get a 'hard' polynomial. With such a high degree we cannot apply [30] to transform depth-4 (say in Theorem 20) to depth-3 in polynomial-time.

Proof of Theorem 4. Let a be an arbitrarily slow growing function. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of polynomials computed by size-s $\Sigma \wedge^a \Sigma\Pi$ circuits that depend only on the first n variables. For all $s \in \mathbb{N}$, let $\mathcal{T}_s$ be the set of polynomials computed by size-s $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables. We show that $\big(\text{poly}(s), O(s^{n/2}/\log^2 s)\big)$-hsg for $\mathcal{P}_s$ or $\mathcal{T}_s$ gives the conclusion of Theorem 4.

Using the hsg for $\mathcal{P}_s$, Theorem 20 gives an efficient variable reduction and Conjecture 1.

Using the hsg for $\mathcal{T}_s$, Theorem 22 gives an efficient variable reduction and Conjecture 1.

After the variable reduction, if we apply hsg of the trivial PIT derandomization [64], we get an $(sd)^{O(\log(sd))}$ time hsg.

To see that the original statement could be proved for *any* field $\mathbb{F}$: Observe that 'depth-4 reduction' [7, Thm.3.2] works for any field. Similarly, we get versions of Theorems 19 & 20 using $\Sigma\Pi^a\Sigma\Pi$ in the respective hypothesis. Also, see the remarks after the proofs of Lemma 12 and Theorem 1. □

5.1 Depth-3 Fpt-blackbox PIT

In this section, we show that we merely need an fpt-algorithm (wrt parameters n, d) for polynomials computed by depth-3 circuits. In *fpt-algorithm*, one provides input with multiple parameters, with the intention that the running time will be polynomial in input size but possibly exponential (or worse) in other parameters [14]. We show that to get the same conclusion as Theorem 4, we merely need a fpt-blackbox PIT for depth-3 circuits computing *multilinear* polynomials. These polynomials have three important complexity parameters: **1)** s, the size of the depth-3 circuit computing the polynomial, **2)** m, the number of variables, and **3)** d, the degree of the polynomial which is upper bounded by m. Here, circuit is the input. So, the running time of the fpt-blackbox PIT must depend polynomially on s. We consider m and d as the extra parameters of the fpt-blackbox PIT. Next, we describe the desired dependence on them.

Theorem 23 (Depth-3 tiny variables & degree). *Suppose that we have a poly(2^{m+d},s)-time computable $\big(2^m + 4^d + s^2\big)/\log^2 s$ degree hsg for m-variate, degree-d* multilinear *polynomials computed by size-s depth-3 circuits.*

Then, we get all the conclusions of Theorem 19.

Remark. 1) Since exponential dependence on m, d is allowed, one can hope that designing such an hsg would be easier than the numerous unsolved PIT cases that the community has attempted till date.

2) Another width-2 ABP version is stated in Theorem 24 (with a worse dependence on m).

3) The number of monomials is 2^m. Thus, the hsg design challenge in the hypothesis of Theorem 23 is barely "above" triviality!

Proof. The proof strategy is identical to that of Theorem 21 and is available in the full version linked on the first page. □

5.2 Log-variate Width-2 ABP or Depth-3 Circuit

A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ has a size-s *width-2 algebraic branching program* (ABP) if it is the $(1,1)$-th entry in the product of s 2×2 matrices (having entries in $\mathbb{F} \cup \{x_i | i\}$).

Theorem 24 (Log-variate width-2 ABP). *Suppose that for some constant $e \geq 1$, we have a $\big(poly(s), O(s^e)\big)$-hsg for polynomials (resp. $2^{e+1} \log s$-degree polynomials) computed by size-s width-2 upper-triangular ABP (resp. depth-3 circuit) that depend only on the first $\log s$ variables. Then, we get all the conclusions of Theorem 19.*

For proof, see the full version linked on the first page.

6 CONCLUSION

We discover the phenomenon of 'efficient bootstrapping' a partial hitting-set generator to a complete one for poly-degree circuits. This inspires a plethora of circuit models. In particular, we introduce the tiny variable diagonal depth-4 (resp. tiny variants of depth-3, width-2 ABP and preprocessed depth-3) model with the motivation that its poly-time hitting-set would: **(1)** solve VP PIT (in quasipoly-time) via a poly-time variable-reducing polynomial map ($n \mapsto \log sd$), and **(2)** prove that either E$\not\subseteq$#P/poly or VNP has polynomials of algebraic complexity $2^{\Omega(n)}$.

Since now we could focus solely on the PIT of VP circuits that depend only on the first *sub*-log (or even constant!) many variables, we need to initiate a study of properties that are useful in that regime. Furthermore, we only need to optimize the size of the hitting-set (& not its time). This work throws up a host of tantalizing models and poses several interesting questions:

- Could the bootstrapping property in Theorem 1 be improved (say, to the function $\log^\star s$)?
- Could the constant parameters in Theorems 2 & 3 be improved? In particular, does $s^{o(n)}$-time blackbox PIT suffice in the latter hypothesis?
- Could we show that the g in Corollary 6 is in VNP and not merely E-computable? This would tie blackbox PIT tightly with the question VNP≠VP (& we can drop 'E$\not\subseteq$# P/poly'). This will require starting with a more structured hsg, so that its annihilator g is a polynomial whose coefficient bits are (#P/poly)-computable. Numerous examples of such polynomials, arising from basic hitting-set designs, can be found in [2, 43] and [41, Sec.4].
- Could we solve *whitebox* PIT for $\log^\star s$ variate (or degree) models? Could it be bootstrapped?
- Could we prove nontrivial lower bounds against the tiny variable (or degree) models?
- Could we solve PIT for n-variate degree-s size-s circuits in $s^{O(\sqrt{n})}$-time?
- Is there a poly(s)-time computable, $O(s^3)$-size hitting-set for 6-variate size-s $\Sigma\Pi\Sigma\wedge$ polynomials?

- An $s^{\exp(n)}$-time computable, $O(s^{n/2})$-size hitting-set for size-s $\Sigma\Pi\Sigma(n)$?
- Could we do blackbox PIT for tiny variable ROABP? For instance, given oracle $C = \sum_{i\in[k]} \prod_{j\in[n]} f_{i,j}(x_j)$ of size$\leq s$, we want a poly$(s,\mu(n))$-hsg, for some μ. It is known that diagonal depth-3 blackbox PIT reduces to this problem if we demand $\mu(n) \leq 2^{O(n)}$ [21].

  Note that for n-variate size-s ROABPs, $s^{O(\log n)}$-time hsg is already known [3]. But, we can ask the following open questions:
- Efficient blackbox PIT for size-s, $\log s$-variate, individual-degree-$(\log^\star s)$ ROABPs?
- Blackbox PIT for size-s, $(\log^\star s) \log s$-variate, multilinear ROABPs?
- Blackbox PIT for size-s, $(\log^\star s) \log s$-variate, $\log s$-degree, diagonal depth-3 circuits? Recently, [20, Thm. 9] gives a polynomial time blackbox PIT algorithm for log-variate depth-3 diagonal circuits.

A PROOFS FROM SECTION 1

Theorem 0 (RESTATED). If we have poly(s)-time blackbox PIT for size-s degree-s circuits, then Conjecture 1 holds.

Proof sketch. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of polynomials computed by size-s degree-s circuits. Using basic linear algebra, we can construct an m-variate multilinear annihilator q_m, where $m = O(\log s)$, for the hsg of $\mathcal{P}_s$ in $2^{O(m)}$-time. This q_m cannot lie in $\mathcal{P}_s$, otherwise q_m evaluated at hsg would be a nonzero polynomial (contradicting the annihilation property). For details, see the proof of [1, Thm.51]. For the sake of contradiction, assume that it has a circuit of size $s^{o(1)}$. Since the degree of q_m is $O(\log s)$, we can invoke the structural log-depth reduction property (see [59, Thm.5.15]) and get a $s^{o(1)}$-size $O(\log s)$-degree circuit computing q_m. Whence $q_m \in \mathcal{P}_s$, which is a contradiction. So we have the polynomial family $\{q_m\}_{m\geq 1}$ such that its coefficients are computable in $2^{O(m)}$-time (thus E-computable) but the algebraic circuit complexity is $> s^{\Omega(1)} = 2^{\Omega(m)}$.

Wlog we assume q_m to be a multilinear polynomial family with 0/1 coefficients; as, indexing a bit of a coefficient requires $O(m)$ bits and one can use the variable-increasing transformation from the proof of [42, Lem.3.9]. Also, if the coefficient function of a polynomial family is in #P/poly, then the polynomial family is in VNP [11, Prop.2.20]. So, if we assume $E \subseteq$ #P/poly, then $\{q_m\}_m$ is in VNP. Thus, either $E \nsubseteq$ #P/poly or VNP has a polynomial family $\{q_m\}_m$ of algebraic circuit complexity $2^{\Omega(m)}$. □

ACKNOWLEDGMENTS

We thank Michael Forbes, Ramprasad Saptharishi, Pranjal Dutta, Thomas Thierauf and Nikhil Balaji for many useful discussions. N.S. thanks the organizers of algebraic complexity workshops in 2014 (MPI Saarbrücken & TIFR Mumbai) that initiated the early discussions with Michael Forbes related to log-variate diagonal depth-3 PIT. S.G. & N.S. thank the organizers, and participants, of the IMSc Chennai workshop in 2017 for discussing the preliminary results. Thanks are due to the anonymous reviewers for giving numerous suggestions to improve an earlier draft. N.S. thanks the funding support from DST (DST/SJF/MSA-01/2013-14). S.G. thanks Google India for providing complete travel expenses to attend the conference.

REFERENCES

[1] Manindra Agrawal. 2005. Proving Lower Bounds Via Pseudo-random Generators. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings.* 92–105.

[2] Manindra Agrawal. 2011. On the Arithmetic Complexity of Euler Function. In *Proceedings of the 6th International Computer Science Symposium in Russia, LNCS 6651.* 43–49. http://www.imsc.res.in/~crs/icm2010conf/talks/manindraslides.pdf

[3] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. 2015. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM J. Comput.* 44, 3 (2015), 669–697.

[4] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. PRIMES is in P. *Annals of mathematics* (2004), 781–793.

[5] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. 2012. Jacobian hits circuits: hitting-sets, lower bounds for depth-D occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC.* 599–614.

[6] Manindra Agrawal, Chandan Saha, and Nitin Saxena. 2013. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013.* 321–330.

[7] Manindra Agrawal and V. Vinay. 2008. Arithmetic Circuits: A Chasm at Depth Four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA.* 67–75.

[8] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, New York, NY, USA.

[9] M. Beecken, J. Mittmann, and N. Saxena. 2013. Algebraic Independence and Blackbox Identity Testing. *Inf. Comput.* 222 (2013), 2–19. (Conference version in ICALP 2011).

[10] Peter Bürgisser. 2001. The complexity of factors of multivariate polynomials. In *In Proc. 42th IEEE Symp. on Foundations of Comp. Science.*

[11] Peter Bürgisser. 2013. *Completeness and reduction in algebraic complexity theory.* Vol. 7. Springer Science & Business Media.

[12] Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. 1991. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science* 84, 2 (1991), 151–164.

[13] Richard A. Demillo and Richard J. Lipton. 1978. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.* 7, 4 (1978), 193 – 195.

[14] Rodney G Downey and Michael R Fellows. 2013. *Fundamentals of parameterized complexity.* Vol. 4. Springer.

[15] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. 2017. *Discovering the roots: Uniform closure results for algebraic classes under factoring.* Technical Report. https://www.cse.iitk.ac.in/users/nitin/research.html. (To appear in 50th ACM Symposium on Theory of Computing (STOC), 2018).

[16] Zeev Dvir, Rafael Mendes de Oliveira, and Amir Shpilka. 2014. Testing Equivalence of Polynomials under Shifts. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming, Part I (Lecture Notes in Computer Science),* Vol. 8572. Springer International Publishing, 417–428. https://doi.org/10.1007/978-3-662-43948-7_35

[17] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. 2016. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.* 754–763.

[18] Ismor Fischer. 1994. Sums of like powers of multivariate linear forms. *Mathematics Magazine* 67, 1 (1994), 59–61.

[19] Michael A Forbes. 2015. Deterministic divisibility testing via shifted partial derivatives. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on.* IEEE, 451–465.

[20] Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. 2018. *Towards blackbox identity testing of log-variate circuits.* Technical Report. https://www.cse.iitk.ac.in/users/nitin/research.html.

[21] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. 2014. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Symposium on Theory of Computing (STOC), New York, NY, USA, May 31 - June 03, 2014.* 867–875.

[22] Michael A. Forbes and Amir Shpilka. 2012. On identity testing of tensors, low-rank recovery and compressed sensing. In *STOC.* 163–172.

[23] Michael A. Forbes and Amir Shpilka. 2017. A PSPACE Construction of a Hitting Set for the Closure of Small Algebraic Circuits. *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), 163. (To appear in 50th ACM Symposium on Theory of Computing (STOC), 2018).

[24] Michael A Forbes, Amir Shpilka, and Ben Lee Volk. 2017. Succinct hitting sets and barriers to proving algebraic circuits lower bounds. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing.* ACM, 653–664.

[25] Martin Fürer. 2009. Faster integer multiplication. *SIAM J. Comput.* 39, 3 (2009), 979–1005.

[26] Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. 2016. A Deterministic Polynomial Time Algorithm for Non-commutative Rational Identity Testing. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016*. 109–117.

[27] Joshua A Grochow. 2015. Unifying known lower bounds via geometric complexity theory. *computational complexity* 24, 2 (2015), 393–475.

[28] Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. 2016. Boundaries of VP and VNP. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 55. 34:1–34:14.

[29] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. 2018. Algebraic dependencies and PSPACE algorithms in approximative complexity. *Electronic Colloquium on Computational Complexity (ECCC)* 25 (2018), 19.

[30] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. 2013. Arithmetic Circuits: A Chasm at Depth Three. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. 578–587.

[31] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. 2017. Identity Testing for Constant-Width, and Any-Order, Read-Once Oblivious Arithmetic Branching Programs. *Theory of Computing* 13, 2 (2017), 1–21. (Preliminary version in CCC'16).

[32] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. 2016. Deterministic Identity Testing for Sum of Read-Once Oblivious Arithmetic Branching Programs. *Computational Complexity* (2016), 1–46. (Conference version in CCC 2015).

[33] Rohit Gurjar and Thomas Thierauf. 2016. Linear Matroid Intersection is in quasi-NC. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 182. (To appear in STOC'17).

[34] Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. 2017. Isolating a Vertex via Lattices: Polytopes with Totally Unimodular Faces. *CoRR* abs/1708.02222 (2017).

[35] Joos Heintz and Claus-Peter Schnorr. 1980. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*. 262–272.

[36] Russell Impagliazzo and Avi Wigderson. 1997. P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC '97)*. ACM, New York, NY, USA, 220–229. https://doi.org/10.1145/258533.258590

[37] Valentine Kabanets and Russell Impagliazzo. 2003. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing (STOC '03)*. 355–364.

[38] Valentine Kabanets and Russell Impagliazzo. 2004. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity* 13, 1-2 (2004), 1–46. https://doi.org/10.1007/s00037-004-0182-6

[39] Erich Kaltofen. 1989. Factorization of polynomials given by straight-line programs. *Randomness and Computation* 5 (1989), 375–412.

[40] Neeraj Kayal and Nitin Saxena. 2007. Polynomial Identity Testing for Depth 3 Circuits. *Computational Complexity* 16, 2 (2007), 115–138.

[41] Pascal Koiran. 2011. Shallow circuits with high-powered inputs. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*. 309–320.

[42] Pascal Koiran and Sylvain Perifel. 2009. VPSPACE and a Transfer Theorem over the Reals. *Computational Complexity* 18, 4 (2009), 551–575.

[43] Pascal Koiran and Sylvain Perifel. 2011. Interpolation in Valiant's Theory. *Computational Complexity* 20, 1 (2011), 1–20.

[44] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. 2014. Equivalence of Polynomial Identity Testing and Deterministic Multivariate Polynomial Factorization. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*. 169–180.

[45] Mrinal Kumar and Shubhangi Saraf. 2016. Arithmetic Circuits with Locally Low Algebraic Rank. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*. 34:1–34:27.

[46] Mrinal Kumar and Shubhangi Saraf. 2016. Sums of Products of Polynomials in Few Variables: Lower Bounds and Polynomial Identity Testing. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*. 35:1–35:29.

[47] Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. 2017. Lower Bounds and PIT for Non-Commutative Arithmetic circuits with Restricted Parse Trees. *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), 77. (Appeared in MFCS'17).

[48] Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. 2016. Non-commutative computations: lower bounds and polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 94.

[49] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*. ACM, 296–303.

[50] Daniel Minahan and Ilya Volkovich. 2017. Complete derandomization of identity testing and reconstruction of read-once formulas. In *LIPIcs-Leibniz International Proceedings in Informatics, CCC'17*, Vol. 79. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[51] Partha Mukhopadhyay. 2016. Depth-4 identity testing and Noether's normalization lemma. In *International Computer Science Symposium in Russia*. Springer, 309–323.

[52] Ketan Mulmuley. 2017. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society* 30, 1 (2017), 225–309.

[53] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. 1987. Matching is As Easy As Matrix Inversion. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*. 345–354.

[54] Ketan D Mulmuley. 2012. The GCT program toward the P vs. NP problem. *Commun. ACM* 55, 6 (2012), 98–107.

[55] Ketan D. Mulmuley. 2012. Geometric Complexity Theory V: Equivalence between Blackbox Derandomization of Polynomial Identity Testing and Derandomization of Noether's Normalization Lemma. In *FOCS*. 629–638.

[56] Noam Nisan and Avi Wigderson. 1994. Hardness vs Randomness. *J. Comput. Syst. Sci.* 49, 2 (1994), 149–167.

[57] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. 2016. Algebraic Independence over Positive Characteristic: New Criterion and Applications to Locally Low Algebraic Rank Circuits. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*. 74:1–74:15. (In print, Computational Complexity, 2018).

[58] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. 2013. A Case of Depth-3 Identity Testing, Sparse Factorization and Duality. *Computational Complexity* 22, 1 (2013), 39–69.

[59] Ramprasad Saptharishi. 2016. *A survey of lower bounds in arithmetic circuit complexity*. Technical Report. https://github.com/dasarpmar/lowerbounds-survey/.

[60] Nitin Saxena. 2008. Diagonal Circuit Identity Testing and Lower Bounds. In *ICALP* (2008-07-08) *(Lecture Notes in Computer Science)*, Vol. 5125. Springer, 60–71.

[61] Nitin Saxena. 2009. Progress on Polynomial Identity Testing. *Bulletin of the EATCS* 99 (2009), 49–79.

[62] Nitin Saxena. 2013. Progress on Polynomial Identity Testing - II. *Electronic Colloquium on Computational Complexity (ECCC)* 20 (2013), 186. http://eccc.hpi-web.de/report/2013/186

[63] Nitin Saxena and C. Seshadhri. 2012. Blackbox Identity Testing for Bounded Top-Fanin Depth-3 Circuits: The Field Doesn't Matter. *SIAM J. Comput.* 41, 5 (2012), 1285–1298.

[64] J. T. Schwartz. 1980. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM* 27, 4 (Oct. 1980), 701–717.

[65] Amir Shpilka and Amir Yehudayoff. 2010. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science* 5, 3-4 (2010), 207–388.

[66] Ola Svensson and Jakub Tarnawski. 2017. The Matching Problem in General Graphs is in Quasi-NC. *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), 59. (Appeared in FOCS'17).

[67] Leslie G. Valiant. 1979. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*. 249–261.

[68] Avi Wigderson. 2017. Low-depth arithmetic circuits: technical perspective. *Commun. ACM* 60, 6 (2017), 91–92.

[69] Andrew C Yao. 1982. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. 23rd Annual Symposium on*. IEEE, 80–91.

[70] Richard Zippel. 1979. Probabilistic Algorithms for Sparse Polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (EUROSAM '79)*. 216–226.