Bootstrapping variables in algebraic circuits

Manindra Agrawal[a,1], Sumanta Ghosh[a,1], and Nitin Saxena[a,1]

[a]Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India

This contribution is part of the special series of Inaugural Articles by members of the National Academy of Sciences elected in 2015.

Contributed by Manindra Agrawal, March 12, 2019 (sent for review January 31, 2019; reviewed by Mrinal Kumar and Srikanth Srinivasan)

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 needs only to consider size-*s* degree-*s* circuits that depend on the first $\log^{\circ c} s$ variables (where *c* is a constant and composes a logarithm with itself *c* times). Thus, the 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 w.r.t. variables. This is repeated *c* times in an efficient way. Pushing the envelope further we show that (*i*) a quadratic-time blackbox PIT for 6,913-variate degree-*s* size-*s* polynomials will lead to a "near"-complete derandomization of PIT and (*ii*) 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 polydegree circuits and implies a lower bound that is a bit stronger than that of Kabanets and Impagliazzo [Kabanets V, Impagliazzo R (2003) *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing STOC '03*].

hitting-set | depth-4 | derandomization | identity testing | lower bound

The 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 an output node; source nodes are called input nodes and are labeled by variables or field constants; noninput 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 (*i*) size, number of edges and vertices (including the variables); (*ii*) depth, number of layers; and (*iii*) degree, maximum degree among all polynomials computed at each node. (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 with its size. So, by "expanding out" the polynomial from the input circuit, we cannot solve the 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 (1–3). 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, e.g., refs. 4–8. The famous algebraic analog is the VP≠VNP question (9). The PIT problem has applications both in proving circuit lower bounds (10–12) and in algorithm design (13–16). For more details on PIT, see the surveys (17–19) or review articles (20, 21).

PIT algorithms are of two kinds: (*i*) whitebox, use the internal structure of the circuit; and (*ii*) blackbox, evaluation of the circuit is allowed only 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 contains 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 an 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 an hsg is a pseudorandom generator (prg) that stretches a seed by several bits or the s extender that stretches n by a single bit. In ref. 22, sections 2 and 3 it is shown 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 an $(\epsilon \log s)$-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.

Significance

Zero testing [or polynomial identity testing (PIT)] for circuits is a computational algebra problem with numerous practical applications and a beautiful theory (e.g., primality testing and graph matching are solved using PIT). It strongly relates to showing that there are explicit/natural polynomials that require exponentially large circuits. The latter is also called the algebraic version of the P≠NP question (or VP≠?VNP) and lies in the intersection of mathematics and computing. This work provides a plausible route to it if we can design a polynomial-time computable, but small enough, hitting set for trivariate depth-4 circuits [merely $\Sigma\Pi\Sigma\wedge(3)$].

Author contributions: M.A., S.G., and N.S. designed research, performed research, contributed new reagents/analytic tools, and wrote the paper.

Reviewers: M.K., University of Toronto; and S.S., Indian Institute of Technology Bombay.

The authors declare no conflict of interest.

Published under the PNAS license.

[1]To whom correspondence may be addressed. Email: manindra@cse.iitk.ac.in, sumghosh@cse.iitk.ac.in, or nitin@cse.iitk.ac.in.

Published online April 11, 2019.

We close this gap in knowledge by showing that an hsg for a size-s degree-s $(\log^{\circ c} s)$-variate circuit can be efficiently converted to an hsg for a size-s degree-s s-variate circuit, where $\log^{\circ c} s := \log \cdots (c\ \mathit{times}) \cdots \log s$. 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 $\delta < 1/2$ and all s) implies "nearly" complete derandomization ($= s^{\exp \circ \exp(O(\log^{\star} s))}$ time hsg for size-s degree-s s-variate circuits). The required $s^{n^{\delta}}$-time PIT is quite close to the trivial s^{n}-time PIT.

We prove an additional result for shallow circuits: Poly(s)-time computable and $O(s^{n/2}/\log^{2} s)$ degree hsg for size-s n-variate depth-4 circuits (for some constant $n \geq 3$) implies quasipolynomial time blackbox PIT for size-s degree-s s-variate circuits (and 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 (23–25), subclasses of depth-4 circuits (26–32), read-once algebraic branching programs (ROABP) and related models (33–39), and certain symbolic determinants (40–43), as well as noncommutative models (44–46). An equally large number of special models have been used to prove lower bounds, for example the ongoing online survey of Saptharishi (47). Also, blackbox PIT relates to conjectures that bar certain algebraic circuit lower-bound methods (48).

Our Notation

$[n]$ refers to $\{1, 2, \ldots, n\}$. Logarithms are w.r.t. 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 (*Theorems 1–4*) hold for any field. However, the other theorems require the 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 $\mathbb{F}[\mathbf{x}]$ (ref. 49, definition 3.1). Efficient hitting sets w.r.t. $\overline{\text{size}}$ are equivalent to an explicit system of parameters (esop) of the invariant ring of a related variety $\Delta[\det(X), s]$ with a given group action (ref. 4, theorem 4.9). Our work (*Theorem 4*) implies that to prove the existence of such a (quasi-)esop it suffices to study the esop w.r.t. X that depend on "constantly few" variables (see the reduction of the derandomized Noether normalization problem NNL to blackbox PIT in ref. 4, section 4.3).

A basic algebraic algorithm used in our results is circuit factoring that relies on field properties. A classic result is ref. 50 that constructs small circuits for factors that have multiplicity coprime to the characteristic (see refs. 51 and 52 for recent factoring results and the related rich background).

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 (*i*) for any nonzero $P \in \mathcal{P}$, $P(\mathbf{f}(y)) \neq 0$, and (*ii*) $\mathbf{f}$ 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 a size-s circuits family is in PSPACE; however, that for a $\overline{\text{size}}$-s circuits family is in EXPSPACE [recently brought down to PSPACE (53, 54)]. The major open question is to bring this complexity down to P; this is christened the "GCT Chasm" in ref. 4, section 11 and has since 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 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 the degree of each variable in f_n is less than δ. For, e.g., 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.

Our Motivation and Main Results

The prg is a well-studied object in Boolean circuit complexity theory and cryptography (refs. 55 and 56, chap. 10). One of the main motivations of studying the 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 an optimal prg against P/poly is equivalent to finding an E-computable Boolean function which has Boolean circuit complexity $2^{\Omega(n)}$ (ref. 22, sections 2.5 and 3.1).

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 (ref. 57, proposition 2.20) if one replaces E by #P/poly, then we are directly talking about a strong version of VNP$\neq$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 $E \nsubseteq$ #P/poly or VNP has a polynomial family of algebraic circuit complexity $2^{\Omega(n)}$.*

In the world of algebraic circuits, the hsg is in direct analogy with the prg. So one can naturally ask about the relation between hsg and algebraic circuit lower bound. Heintz and Schnorr (ref. 10, theorem 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 the 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.*

A weak converse of *Theorem 0*, i.e., hardness to hsg, is well known due to ref. 58, theorem 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, we apply a "trivial" hitting set on the $O(\log s)$ variables, which gives a quasi-polynomial time hsg for $\mathcal{P}_s$ (59). This suggests that the "hardness vs. randomness" connection here is less satisfactory than the Boolean world. Nonetheless, we wonder 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 an hsg for $\mathcal{P}_s$. Clearly, $\mathbf{g}_{s^c,s}$ can be computed in poly(s) time. However, for $m \le s^{o(1)}$, 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 with that of the latter "much larger" polynomial ring $\mathbb{F}[x_1, \ldots, x_s]$; so in the underlying algebraic-geometry concepts a terrible blowup is warranted.

For any $c \in \mathbb{N}$, let $\log^{\circ c}$ be defined as c-times application of the base-2 logarithm function (e.g., $\log^{\circ 3} s = \log\log\log s$). 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 (e.g., $m = \log^\star s$), 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: 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 $< s$.

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. Although we do not get an equivalence result now, we do, however, weaken the hypothesis very significantly.

Theorem 2. *Suppose, for constants $e \ge 2$ and $1 > \epsilon \ge (3 + 6\log(128e^2))/(128e^2)$, 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 $\epsilon = 6{,}912/6{,}913$, then the hypothesis required is quadratic-time [i.e., $O(s^2)$] blackbox PIT for 6,913-variate degree-s size-s polynomials.

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

Remarks:

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

ii) 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 layers to mere powering gates (24, 60). Indeed, we can prove stronger versions of *Theorem 4*: for $\Sigma\wedge\Sigma\Pi$ (*Theorem 19*) resp. $\Sigma\Pi\Sigma\wedge$ (*Theorem 21*) circuits in the hypothesis. But, these results (unlike *Theorems 1–4*) require the field characteristic to be zero or large.

iii) Our conclusion is as strong as those obtained via the well-known "constant-depth reduction" results in refs. 60 and 61. But our hypothesis needs an hsg only slightly better than trivial.

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 the lower bound. The latter will follow from *Theorem 0*, so we describe only the proof idea of the 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 $0 \le i \le c+1$, $f_i(s) := (\log^{\circ i} s)^2$. Using induction, we show that for $0 \le i \le c+1$, 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 *Theorem 1*. 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:

i) 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 the 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 the 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).

ii) 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.

iii) 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 ref. 58, theorem 7.7 as it can handle hard nonmultilinear 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*, we can get such a polynomial family (as in the proof of *Theorem 0* but now the hard polynomial will be nonmultilinear). In step *i* above, we are working with a 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 *Lemma 12* 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 the NW design for each exponential stretch of variables; we do not know how to directly get a hyperexponential stretch and save 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 $n = 6{,}913$ (or, for $e = 3$, $n = 17{,}574$ 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 nonconstant w.r.t. s. We show that if we have an s^{f_i}-time hsg, in the ith induction step, then in the next step we get an $s^{f_{i+1}}$-time hsg, where $f_{i+1} := 16 f_i^2$. So, after $\log^\star s$ steps, we get an hsg of our desired complexity (equal to slightly superpolynomial).

Like *Lemma 12*, here *Lemma 17* combines all of 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. *Theorems 2* and *3* 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 (and are still open)—integer multiplication (62) and matrix multiplication (63).

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, where $a(s)$ 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 the (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* (*Theorem 19* and resp. ref. 21). 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 the PIT part. Next, we give the proof sketch of the variable reduction part.

First, we discuss the variable reduction part assuming the $O(s^{n/2}/\log^2 s)$-degree hsg of $\mathcal{P}_s$. 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 a (poly(s), $O(s^n/\log^2 s)$)-hsg of $\mathcal{P}'_s$.

Since the number of $\frac{n}{2}\log s$-degree multilinear monomials over $m := n\log s$ variables is $\binom{m}{m/2} \geq 2^m/\sqrt{2m} = s^n/\sqrt{2m} > O(s^n/\log^2 s) \cdot m$ (for large enough s), 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*, the 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 the depth reduction of ref. 61, 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 the Kronecker map $(x_{u(j)+i} \mapsto y_j^{2^i},\ i \in [\log s])$ with the jth block defined as $\{x_{u(j)+i} \mid i \in [\log s]\}$ for an appropriate $u(\cdot)$. 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 19*.

Second, we discuss the variable reduction part assuming an efficient $O(s^{n/2}/\log^2 s)$-degree hsg of $\mathcal{T}_s$. The proof idea is similar to the previous one; the 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 that in ref. 60. For details see *Theorems 20* and *21*.

Brushing Up Relevant Techniques

In this section we revisit the techniques that have appeared in some form in refs. 10–12, 22, 60, and 61.

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 ref. 10, theorem 4.5 and ref. 12, theorem 51. In *Lemma 5*, we prove a revised version. Later, it is used to get a hard polynomial from hitting-set generator.

Lemma 5 (Hitting Set 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 $\mathrm{poly}(tdn)$-time, has individual degree less than $\delta := \lceil d^{3/n} \rceil$, and is homogeneous of degree $(\delta - 1)n/2$.*

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.*

Toward a converse of *Lemma 5*, a crucial ingredient is the Nisan–Wigderson design (22). 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, ref. 56, chap.16). *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 13 improves this.)*

Lemma 9 is a revised version of the counterpositive of ref. 58, 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.

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).

Lemma 10 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 (9).

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

Lemma 11 converts a monomial into a sum of powers. It is called Fischer's trick in ref. 60. It requires char $\mathbb{F} = 0$ or large.

Lemma 11 [Fischer's Trick (64)]. *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 $\{g_{i'j} \mid j\}$ for some i'.*

Perfect Bootstrapping—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 the number of variables can be as low as $\log^{\circ c} s$ compared with s, the size of the circuit. In this extremely low-variate regime, we have to deal with a hard polynomial family of nonconstant 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 is 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 (ref. 57, theorem 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 follows: 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 $P(p_1, \ldots, p_n)$ is zero. Since $P(\mathbf{x})$ is nonzero, we can find the smallest $j \in [n]$ such that $P(p_1, \ldots, p_{j-1}, x_j, \ldots, x_n) =: P_1$ is nonzero, but $P_1|_{x_j = p_j}$ is zero. Thus, $(x_j - p_j)$ divides P_1. Let $\mathbf{a}$ be a constant assignment on all of the variables in P_1, except x_j and the variables S_j in p_j, with the property that P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, we can find such an assignment (3). Now our new polynomial P_2 on the variables S_j and x_j is of the form $P_2(S_j, x_j) = P(p'_1, \ldots, p'_{j-1}, x_j, a_{j+1}, \ldots, a_n)$, where for each $i \in [j-1]$, p'_i is the polynomial on the variables $S_i \cap S_j$, and the a_is are field constants decided by our assignment $\mathbf{a}$. By the design, for each $i \in [j-1]$, $|S_i \cap S_j| \leq 10\varepsilon(s'^{c_0})$. Since p'_i are polynomials on variables $S_i \cap S_j$ of individual degree $\leq \delta$, 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 (50) (see ref. 57, theorem 2.21 for the algebraic circuit complexity of factors) and get an algebraic circuit for p_j of size $(s_1 d_1)^{c_3}$

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

This exponent $= \left(\frac{3}{\lceil(9\sqrt{e}+3)\rceil} + \frac{30e}{\lceil 30e+10\sqrt{e+1}\rceil}\right)c_0 \le \left(\frac{1}{(3\sqrt{e}+1)} + \frac{3\sqrt{e}}{3\sqrt{e}+\sqrt{1+1/e}}\right)c_0 < c_0$. 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 (*i*) computing the design [i.e., poly($2^{m'}$) time] and (*ii*) 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 $< nd \cdot \deg(q) = \text{poly}(sd)$. □

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, as in ref. 58, remark 7.5, to redefine algebraic complexity of q over $\mathbb{F}_{r^t}$ suitably and *Lemma 12* 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 (ref. 47, theorem 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 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 $\mathcal{P}_{s,d,c+1}$ 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 $i \in [c+1]$.

i) 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, the 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 (*i*) the individual degree is less than $\delta_{m,s} = \lceil s^{3e/m} \rceil$, (*ii*) $q_{m,s} \notin \mathcal{T}_{s,i}$, and (*iii*) it is 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.

ii) 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 *Lemma 12* (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 a poly(sd)-size circuit which depends only on the first $c_2\varepsilon_i((sd)^{c_0})$ variables.

iii) 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 the polytime hsg of $\mathcal{P}_{s',d',i}$ we get a polytime 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 $\{q_m\}_{m \ge 1}$ such that (*i*) it is multi-δ-ic, for some constant δ, and (*ii*) it is 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 *i* of the induction step above. As remarked before, we can define $\mathcal{T}_{s,i}$ to be the set of polynomials $f(x_1, \dots, x_{c_1\varepsilon_i(s)})$, 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 *ii*.

Bootstrapping Constant Variate—Proof of *Theorems 2* and *3*

The overall strategy is similar to that in the previous section. However, the details would now change drastically. The technical proofs of this section are in the full version of ref. 65.

First, we describe an optimized version of the NW design, where the parameters are different from those in *Lemma 8*. Later, it will help us improve the constants.

Lemma 13 (NW Design). *There exists an algorithm which takes* (ℓ, n, d), *with* $\ell \ge 100$ *and* $d \ge 13$, *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* $\ge 2^{d/4}$ *subsets, in time* $O((4\ell/n)^n)$.

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

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

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 do this in *Propositions 14* and *15*.

From now on we 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 $i \geq 0$.

Proposition 14. *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 15 (m_i Is a Tetration). *Suppose that $m_0 \geq \max\{(8f_0)^{\frac{2}{\epsilon}},\ 192f_0^2 \cdot \frac{1}{\epsilon}\log(128f_0^2)\}$. Then for all $i \geq 0$, (i) $m_{i+1} \geq 2^{m_i^{1-\epsilon}}$ and (ii) $m_{i+1} \geq 2m_i > 3{,}456f_i^2$.*

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

Proposition 16 (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 ith step of bootstrapping.

Lemma 17 (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 w.r.t. 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 a 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 the same time complexity. Like *Lemma 5*, the annihilator of the hsg of $\mathcal{T}_s$ gives a polynomial q_s such that (*i*) $q_s \notin \mathcal{T}_s$, (*ii*) it is computable in d_i^4 time by linear algebra, and (*iii*) 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 in *Proposition 15* we have $m_i \geq 1{,}728f_i^2$.

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 $m_i \geq 1{,}728f_i^2$, 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 (*i*) q_s is computable in $d_i^4 = s^{4f_i}$ time but has no circuit of size $\leq s$ and (*ii*) 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 nonzero polynomial in $\mathcal{P}_{s,i+1}$ to a nonzero 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, together with *Lemma 13*, ensure that such a design exists. Define for all $j \in [m']$, $p_j := q_{s_0}(S_j)$. Next, we show that for any nonzero $P \in \mathcal{P}_{s,i+1}$, $P(p_1, \ldots, p_{m'})$ is also nonzero.

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 of the variables in P_1, except x_j and the variables S_j in p_j, with the property that P_1 at $\mathbf{a}$ is nonzero. Since P_1 is nonzero, we can find such an assignment (3). 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 the a_is 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 (50) (see ref. 57, theorem 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 $\delta_{s_0} = 1 + s_0^{f_i(8f_i+1)/m_i}$, $m_i = s^{o(1)}$, $m' \leq s$. Let us upper bound $s_0' =$

$$\begin{aligned}
s_1\tilde{O}(d_1^2) &\leq \left(s + \frac{m' m_i}{16f_i^2} \cdot \delta_{s_0}^{1+\frac{m_i}{16f_i^2}}\right) \cdot \tilde{O}\left(\frac{s m_i \delta_{s_0}}{16f_i^2}\right)^2 \\
&\leq \frac{s^{3+o(1)}\delta_{s_0}^2}{f_i^2} + \frac{s^{3+o(1)}\delta_{s_0}^{3+\frac{m_i}{16f_i^2}}}{f_i^4} \\
&\leq s^{3+o(1)} s_0^{\frac{2f_i(8f_i+1)}{m_i}} + s^{3+o(1)} s_0^{\left(3+\frac{m_i}{16f_i^2}\right)\frac{f_i(8f_i+1)}{m_i}} \\
&\leq s^{3+o(1)} s^{\frac{14f_i(8f_i+1)}{m_i}} + s^{3+o(1)} s^{\left(3+\frac{m_i}{16f_i^2}\right)\frac{7f_i(8f_i+1)}{m_i}} \\
&\leq s^{3+o(1)} s^{\frac{112f_i^2+14f_i}{m_i}} + s^{3+o(1)} s^{\frac{21f_i(8f_i+1)}{m_i} + \frac{7(8f_i+1)}{16f_i}} \\
&\leq s^{3+o(1)} s^{\frac{112f_i+14}{1{,}728f_i}} + s^{3+o(1)} s^{\frac{21(8f_i+1)}{1{,}728f_i} + \frac{7(8f_i+1)}{16f_i}} \\
&\quad (\because m_i > 1{,}728f_i^2)
\end{aligned}$$

$$\leq s^{3+o(1)+\frac{112}{1,728}+\frac{7}{1,728}} + s^{3+o(1)+\frac{168}{1,728}+\frac{21}{3,456}+\frac{56}{16}+\frac{7}{32}}$$
$$(\because f_i \geq 2)$$
$$\leq s^{3.1+o(1)} + s^{6.83+o(1)} < s^7 = s_0 \,.$$

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

Using the Given hsg. The above variable reduction converts P to an m_i-variate degree-d' nonzero polynomial P' computable by an s'-size circuit, where

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

Now we give an upper bound of s':

$$\begin{aligned} s' &= s + \frac{m' m_i \delta_{s_0}}{8f_i} \cdot \delta_{s_0}^{\frac{m_i}{8f_i}} \\ &= s + \frac{m' m_i}{8f_i} \cdot \left(1 + s_0^{\frac{f_i(8f_i+1)}{m_i}}\right)^{(\frac{m_i}{8f_i}+1)} \\ &\leq s + \frac{m' m_i}{8f_i} \cdot (1+s)^{\frac{7f_i(8f_i+1)}{m_i}(\frac{m_i}{8f_i}+1)} \\ &\leq s + s^{1+o(1)+7(f_i+\frac{1}{8})(1+\frac{8f_i}{m_i})} \\ &< s + s^{1+o(1)+7(f_i+\frac{1}{8})(1+\frac{8}{3,456})} \quad (\because m_i > 1{,}728 f_i^2, f_i \geq 2) \\ &< s^{9f_i} \,. \end{aligned}$$

Since $d', s' < s^{9f_i} =: s_1$, P' is m_i-variate degree-s_1 polynomial that is computable by a 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 the NW design (by the greedy algorithm sketched in *Lemma 13*) $\{S_1, \ldots, S_{m'}\}$ 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 jth 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 the hsg for P has four components:

i) Computing q_{s_0} (step *b*): It takes $(s_0^{f_i})^4 = s^{7\times f_i \times 4} = s^{28f_i} \leq s^{14f_i^2}$.
ii) Nisan–Wigderson design from *Lemma 13* (step *c*): It takes time $O\left(4m_i/(m_i/8f_i)\right)^{m_i/8f_i} = O(32f_i)^{m_i/8f_i}$. If $m_i > 64f_i^2 \log s$, then we will run the ith induction step only for (relabeled) $m_i := 64f_i^2 \log s$, as the stretch obtained will already be to $2^{m_i/64f_i^2} = s$ variables. Note that at that point, i would be nonconstant and hence $f_i > 4$. In this regime, $(32f_i)^{m_i/8f_i} = (32f_i)^{8f_i \log s} = s^{8f_i \log(32f_i)} < s^{12f_i^2}$.
iii) Computing C (step *d*): Essentially, compute m' copies of q_{s_0} (in dense representation). As seen before, the total time complexity is s^{9f_i}.
iv) Computing the hsg of $\mathcal{P}_{s^{9f_i},i}$ and then computing the hsg of $\mathcal{P}_{s,i+1}$ by composition (step *e*): It takes $s^{9f_i^2} + s^{9f_i^2} \cdot s^{9f_i} < s^{14f_i^2}$ time.

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

Proof of Theorem 2: In the hypothesis of the *Theorem 2* statement we are given constants $e \geq 2$ and $n \geq 1{,}024$. 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 17*) several times to boost m_0 variables to an arbitrary amount.

Let P be a degree-s polynomial computed by a size-s circuit. Then, it can have at most s variables. Let k be the smallest integer such that $m_k \geq s$ (k is an extremely slow-growing function in s as described in *Proposition 16*). By *Proposition 15*, we have that $m_{k-1} \leq s^{o(1)}$.

For $i \in \mathbb{N}_{\geq 0}$ 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 17*, to get the $t'^{f_{i+1}}$-hsg for $\mathcal{P}_{t',i+1}$. After k such applications of *Lemma 17*, we get an s^{f_k}-time hsg for s-variate degree-s polynomials computed by size-s circuits.

Note that $f_k = (16f_0)^{2^k}/16 = 2^{O(2^k)} = 2^{2^{O(\log^\star s)}}$. 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 $\{q_m\}_{m\geq 1}$ 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. Without loss of generality (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 $n := \lceil (64e^2)^{1/\epsilon} \rceil \geq 192e^2 \log(128e^2)^{1/\epsilon}$ 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*. □

Remarks:

i) The NW (ℓ, n, d) design that we are using, in the ith iteration (*Lemma 17*), 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 $1/2$. An optimal result was provided recently by ref. 66.
ii) 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 analyzing the bit complexity (i.e., time) and the algebraic complexity (i.e., degree of the hsg) separately in the proof of *Lemma 17*. We skip the details for now.

Shallow Bootstrapping—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_is are sparse polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $\leq b$, $a_i \leq a$, and c_is 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 fan-in (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 models seem weaker than general depth-4 circuits. So one would expect that solving PIT for these models would be easier.

Our proofs will cover a plethora of models. *Theorems 18* and *19* together give the proof of our first approach. *Theorems 20* and *21* together give the proof of the second approach. One can note 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 quasi-polynomial hsg for size-s degree-d circuits. However, after the variable reduction, if we apply hsg of the trivial PIT derandomization (3), we get an $(sd)^{O(\log(sd))}$-time hsg.

Theorem 18 ($\Sigma\wedge^a\Sigma\Pi$ 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 ref. 61, theorem 3.2.

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 *Conjecture 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*. $\square$

Remarks:

i) 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.

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

iii) 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 nonmultilinear 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 19 (Tiny Variate $\Sigma\wedge^a\Sigma\Pi$). *Suppose that for some constant $n\ge 3$ 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 of the conclusions of Theorem 18.*

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 18*, 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 $\varphi(P)\ne 0$ (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 $\big(\text{poly}(s), O(s^n/\log^2 s)\big)$-hsg for P. $\square$

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

Theorem 20 (Depth-3 Computing Multilinear). *Suppose that for some constant $n\ge 2$, we have a $\big(\text{poly}(s), O(s^n/\log^2 s)\big)$-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 of the conclusions of Theorem 18.*

Proof: Proof is similar to proof of *Theorem 18*. The main difference is that there we are dealing with depth-4 circuits, but here we have depth-3 circuits. So we need a "depth-3 reduction" result (60) with a "depth-4 reduction" result (61). We sketch only 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 $k=\binom{m}{m/2}\ge 2^m/\sqrt{2m}=s^n/\sqrt{2m}>\ O(s^n/\log^2 s)\cdot m$ (for large enough s). Thus, by linear algebra similar to *Lemma 5*, we get an m-variate $m/2$-degree multilinear homogeneous annihilating polynomial $q_m\notin\mathcal{P}_s$ 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 18*, after flattening 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 ref. 60 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 21 (Tiny Variate $\Sigma\Pi\Sigma\wedge$). *Suppose that for some constant $n \geq 3$, we have a $\left(poly(s),\, O(s^{n/2}/\log^2 s)\right)$-hsg for polynomials computed by size-s $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables. Then, we get all of the conclusions of Theorem 18.*

Proof: The proof is similar to that of *Theorem 19*. For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of multilinear polynomials computed by size-s depth-3 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 $\Sigma\Pi\Sigma\wedge$ circuits that depend only on the first n variables. According to the hypothesis, we have an $O(s^{n/2}/\log^2 s)$-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 $O(s^n/\log^2 s)$-degree 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 20*, we get our conclusion.

Now 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 $\Sigma\Pi\Sigma\wedge$ circuit of size at most $s + s \cdot 2^{\log s} = O(s^2)$. Thus, using the $O(s^n/\log^2 s)$-degree hsg for $\mathcal{T}_{O(s^2)}$, we get a $\left(\text{poly}(s),\, O(s^n/\log^2 s)\right)$-hsg for P. ☐

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 (*i*) the above trick of applying the local Kronecker map, to reduce variables from $n \log s$ to n, increases the circuit depth to 4 and moreover, any such map has to be nonlinear, as otherwise the resulting monomials are too few, and (*ii*) 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 ref. 60 to transform depth-4 (say in *Theorem 19*) 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 $\left(\text{poly}(s),\, O(s^{n/2}/\log^2 s)\right)$-hsg for $\mathcal{P}_s$ or $\mathcal{T}_s$ gives the conclusion of *Theorem 4*.

Using the hsg for $\mathcal{P}_s$, *Theorem 19* gives an efficient variable reduction and *Conjecture 1*.

Using the hsg for $\mathcal{T}_s$, *Theorem 21* gives an efficient variable reduction and *Conjecture 1*.

After the variable reduction, if we apply the hsg of the trivial PIT derandomization (3), 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 (ref. 61, theorem 3.2) works for any field. Similarly, we get versions of *Theorems 18* and *19* using $\Sigma\Pi^a\Sigma\Pi$ in the respective hypothesis. Also, see the remarks after the proofs of *Lemma 12* and *Theorem 1*. ☐

Log-Variate Width-2 Algebraic Branching Program 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 $\mathbb{F} \cup \{x_i \mid i\}$).

Theorem 22 (Log-Variate Width-2 ABP). *Suppose that for some constant $e \geq 1$, we have a $(poly(s),\, O(s^e))$-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 of the conclusions of Theorem 18.*

Proof: In ref. 67, theorem 3 an efficient transformation was given that rewrites a size-s depth-3 circuit, times a special product of linear polynomials, as a poly(s)-size width-2 upper-triangular ABP. Thus, an hsg for the latter model gives a similar hsg for the former. So, from the hypothesis for some constant e, we have a (poly(s), $O(s^e)$)-hsg for $2^{e+1}\log s$-degree polynomials computed by size-s depth-3 circuits that depend only on the first $\log s$ variables.

For all $s \in \mathbb{N}$, let $\mathcal{P}_s$ be the set of $\log s$-variate, $2^{e+1}\log s$-degree polynomials computed by size-s depth-3 circuits. Let $d := 2^{e+1}\log s$. Let $\mathbf{f}(y)$ be the (poly(s), $O(s^e)$)-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 $m := \log s$-variate d-degree monomials. Then $k = \binom{m+d-1}{m} > 2^{(e+1)\log s} = s^{e+1}$. Since $k > O(s^e)\cdot d$, we get an m-variate d-degree homogeneous annihilating polynomial $q_m \notin \mathcal{P}_s$ computable in $s^{O(1)}$ time. The analysis is similar to *Lemma 5*. Importantly $q_m \notin \mathcal{P}_s$, and thus no depth-3 circuit of size $< s = 2^{\Theta(m)}$ can compute it.

From this point onward the proof of *Theorem 20* is identical and we are done. ☐

Conclusion

We discover the phenomenon of "efficient bootstrapping" of a partial hitting-set generator to a complete one for polydegree 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 polytime hitting-set would (*i*) solve VP PIT (in quasipoly time) via a polytime variable-reducing polynomial map ($n \mapsto \log sd$) and (*ii*) prove that either E$\not\subseteq$#P/poly or VNP has polynomials of algebraic complexity $2^{\Omega(n)}$. Could the bootstrapping property in *Theorem 1* be improved (say, to the function $\log^\star s$)? Could the constant parameters in *Theorems 2* and *3* be improved? In particular, does $s^{o(n)}$-time blackbox PIT suffice in the latter hypothesis? The latter question has been resolved in ref. 66. Is there efficient blackbox PIT for size-s, $\log s$-variate, individual-degree $(\log^\star s)$ ROABPs? Is there blackbox PIT for size-s, $(\log^\star s)\log s$-variate, multilinear ROABPs? Is there blackbox PIT for size-s, $(\log^\star s)\log s$-variate, $\log s$-degree, diagonal depth-3 circuits? Recently, ref. 68, theorem 9 gave a polynomial-time blackbox PIT algorithm for log-variate depth-3 diagonal circuits.

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 (Max Planck Institute Saarbrücken and Tata Institute of Fundamental Research Mumbai) that initiated the early discussions with Michael Forbes related to log-variate diagonal depth-3 PIT. S.G. and N.S. thank the organizers and participants of the Institute of Mathematical Sciences Chennai workshop in 2017 for discussing the preliminary results. We thank the anonymous reviewers for giving numerous suggestions to improve an earlier draft. S.G. thanks Google India for generously sponsoring the trip to the conference. N.S. acknowledges the funding support from Department of Science & Technology (India) (Award DST/SJF/MSA-01/2013-14), and M.A. thanks J. C. Bose Fellowship. This work was presented in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018* (65).

1. Demillo RA, Lipton RJ (1978) A probabilistic remark on algebraic program testing. *Inf Process Lett* 7:193–195.
2. Zippel R (1979) Probabilistic algorithms for sparse polynomials. *Proceedings of the International Symposium on Symbolic and Algebraic Computation EUROSAM '79* ed Ng EW (Springer, New York), Vol 72, pp 216–226.
3. Schwartz JT (1980) Fast probabilistic algorithms for verification of polynomial identities. *J ACM* 27:701–717.
4. Mulmuley K (2017) Geometric complexity theory V: Efficient algorithms for Noether normalization. *J Am Math Soc* 30:225–309.
5. Grochow JA, Mulmuley KD, Qiao Y (2016) Boundaries of VP and VNP. *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Leibniz International Proceedings in Informatics (LIPIcs), eds Chatzigiannakis I, Mitzenmacher M, Rabani Y, Sangiorgi D (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), Vol 55, pp 34.1–34.14.
6. Grochow JA (2015) Unifying known lower bounds via geometric complexity theory. *Comput Complexity* 24:393–475.
7. Mulmuley KD (2012) Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma. *53rd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, Piscataway, NJ), pp 629–638.
8. Mukhopadhyay P (2016) Depth-4 identity testing and Noether's normalization lemma. *International Computer Science Symposium in Russia*, eds Kulikov AS, Woeginge GJ (Springer, New York), Vol 9691, pp 309–323.
9. Valiant LG (1979) Completeness classes in algebra. *Proceedings of the 11h Annual ACM Symposium on Theory of Computing*, eds Fischer MJ, DeMillo RA, Lynch NA, Burkhard WA, Aho AV (ACM, New York), pp 249–261.
10. Heintz J, Schnorr C-P (1980) Testing polynomials which are easy to compute (extended abstract). *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, eds Miller RE, Ginsburg S, Burkhard WA, Lipton RJ (ACM, New York), pp 262–272.
11. Kabanets V, Impagliazzo R (2003) Derandomizing polynomial identity tests means proving circuit lower bounds. *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing STOC '03*, eds Larmore LL, Goemans MX (ACM, New York), pp 355–364.
12. Agrawal M (2005) Proving lower bounds via pseudo-random generators. *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference December 15-18*, eds Ramanujam R, Sen S (Springer, Berlin), Vol 3821, pp 92–105.
13. Mulmuley K, Vazirani UV, Vazirani VV (1987) Matching is as easy as matrix inversion. *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing STOC '87*, ed Aho AA (ACM, New York), pp 345–3154.
14. Agrawal M, Kayal N, Saxena N (2004) PRIMES is in P. *Ann Math* 160:781–793.
15. Kopparty S, Saraf S, Shpilka A (2014) Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. *IEEE 29th Conference on Computational Complexity CCC* (IEEE, Piscataway, NJ), pp 169–180.
16. Dvir Z, de Oliveira RM, Shpilka A (2014) Testing equivalence of polynomials under shifts. *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming, Part I*, Lecture Notes in Computer Science, eds Esparza J, Fraigniaud P, Husfeldt T, Koutsoupias E (Springer, New York), Vol 8572, pp 417–428.
17. Saxena N (2009) Progress on polynomial identity testing. *Bull EATCS* 99:49–79.
18. Saxena N (2014) Progress on polynomial identity testing–II. *Perspectives in Computational Complexity* (Springer, New York), pp 131–146.
19. Shpilka A, Yehudayoff A (2010) Arithmetic circuits: A survey of recent results and open questions. *Found Trends Theor Comput Sci* 5:207–388.
20. Wigderson A (2017) Low-depth arithmetic circuits: Technical perspective. *Commun ACM* 60:91–92.
21. Mulmuley KD (2012) The GCT program toward the P vs. NP problem. *Commun ACM* 55:98–107.
22. Nisan N, Wigderson A (1994) Hardness vs randomness. *J Comput Syst Sci* 49:149–167.
23. Kayal N, Saxena N (2007) Polynomial identity testing for depth 3 circuits. *Comput Complexity* 16:115–138.
24. Saxena N (2008) Diagonal circuit identity testing and lower bounds. *ICALP*, Lecture Notes in Computer Science, eds Aceto L, Damgård I, Goldberg LA, Halldórsson MM, Ingólfsdóttir A, Walukiewicz I (Springer, Berlin), Vol 5125, pp 60–71.
25. Saxena N, Seshadhri C (2012) Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J Comput* 41:1285–1298.
26. Agrawal M, Saha C, Saptharishi R, Saxena N (2012) Jacobian hits circuits: Hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits, eds Karloff HJ, Pitassi T (ACM, New York), pp 599–614.
27. Beecken M, Mittmann J, Saxena N (2013) Algebraic independence and blackbox identity testing. *Inf Comput* 222:2–19.
28. Saha C, Saptharishi R, Saxena N (2013) A case of depth-3 identity testing, sparse factorization and duality. *Comput Complexity* 22:39–69.
29. Forbes MA (2015) Deterministic divisibility testing via shifted partial derivatives. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, ed Guruswami V (IEEE, Piscataway, NJ), pp 451–465.
30. Kumar M, Saraf S (2016) Arithmetic circuits with locally low algebraic rank.*31st Conference on Computational Complexity CCC*, ed Raz R (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), Vol 50, pp 34:1–34:27.
31. Kumar M, Saraf S (2016) Sums of products of polynomials in few variables: Lower bounds and polynomial identity testing. *31st Conference on Computational Complexity CCC* ed Raz R (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), pp 35:1–35:29.
32. Pandey A, Saxena N, Amit S (2018) Algebraic independence over positive characteristic: New criterion and applications to locally low algebraic rank circuits. *Comput Complex* 27:617–670.
33. Forbes MA, Shpilka A (2012) On identity testing of tensors, low-rank recovery and compressed sensing. *Proceedings of the 44th Symposium on Theory of Computing Conference*, eds Karloff HJ, Pitassi T (ACM, New York), pp 163–172.
34. Agrawal M, Saha C, Saxena N (2013) Quasi-polynomial hitting-set for set-depth-Δ formulas. *Symposium on Theory of Computing Conference STOC'13*, eds Boneh D, Roughgarden T, Feigenbaum J (ACM, New York), pp 321–330.
35. Forbes MA, Saptharishi R, Shpilka A (2014) Hitting sets for multilinear read-once algebraic branching programs, in any order. *Symposium on Theory of Computing (STOC)*, ed Shmoys DB (ACM, New York), pp 867–875.
36. Agrawal M, Gurjar R, Korwar A, Saxena N (2015) Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J Comput* 44:669–697.
37. Gurjar R, Korwar A, Saxena N, Thierauf T (2016) Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *Comput Complexity* 26: 1–46.
38. Gurjar R, Korwar A, Saxena N (2017) Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theor Comput* 13:1–21.
39. Minahan D, Volkovich I (2017) Complete derandomization of identity testing and reconstruction of read-once formulas. *LIPIcs-Leibniz International Proceedings in Informatics, CCC'17*, ed O'Donnell R (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), Vol 79.
40. Fenner SA, Gurjar R, Thierauf T (2016) Bipartite perfect matching is in quasi-NC. *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, eds Wichs D, Mansour Y (ACM, Cambridge, MA), pp 754–763.
41. Gurjar R, Thierauf T (2017) Linear matroid intersection is in quasi-nc. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing STOC*, eds Hatami H, McKenzie P, King V (ACM, New York), pp 821–830.
42. Svensson O, Tarnawski J (2017) The matching problem in general graphs is in quasi-nc. *58th IEEE Annual Symposium on Foundations of Computer Science FOCS 2017*, ed Umans C (IEEE, Piscataway, NJ), pp 696–707.
43. Gurjar R, Thierauf T, Vishnoi NK (2018) Isolating a vertex via lattices: Polytopes with totally unimodular faces. *45th International Colloquium on Automata, Languages, and Programming ICALP*, eds Chatzigiannakis I, Kaklamanis C, Marx D, Sannella D (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), pp 74:1–74:14.
44. Lagarde G, Malod G, Perifel S (2016) Non-commutative computations: Lower bounds and polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)*, Vol 23, pp 1–72.
45. Garg A, Gurvits L, Oliveira R, Wigderson A (2016) A deterministic polynomial time algorithm for non-commutative rational identity testing. *IEEE 57th Annual Symposium on Foundations of Computer Science FOCS*, ed Dinur I (IEEE, Piscataway, NJ), pp 109–117.
46. Lagarde G, Limaye N, Srinivasan S (2017) Lower bounds and PIT for non-commutative arithmetic circuits with restricted parse trees. *42nd International Symposium on Mathematical Foundations of Computer Science MFCS*, eds Larsen KG, Bodlaender HL, Raskin J-F (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), Vol 83, pp 41:1–41:14.
47. Saptharishi R (2016) A survey of lower bounds in arithmetic circuit complexity. Version 3. Available at https://github.com/dasarpmar/lowerbounds-survey/releases. Accessed April 5, 2019.
48. Forbes MA, Shpilka A, Lee Volk B (2017) Succinct hitting sets and barriers to proving algebraic circuits lower bounds. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, eds Hatami H, McKenzie P, King V (ACM, New York), pp 653–664.
49. Bürgisser P (2001) The complexity of factors of multivariate polynomials. *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ).
50. Kaltofen E (1989) Factorization of polynomials given by straight-line programs. *Advances in Computing Research*, Vol 5, pp 375–412.
51. Dutta P, Saxena N, Amit S (2018) Discovering the roots: Uniform closure results for algebraic classes under factoring. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing STOC 2018*, eds Diakonikolas I, Kempe D, Henzinge M (ACM, New York), pp 1152–1165.
52. Dwivedi A, Mittal R, Saxena N (2019) Counting basic-irreducible factors mod p^k in deterministic poly-time and p-adic applications. arXiv:1902.07785. Preprint, posted February 20, 2019.
53. Guo Z, Saxena N, Amit S (2018) Algebraic dependencies and PSPACE algorithms in approximative complexity. *33rd Computational Complexity Conference CCC*, ed Servedio RA (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), pp 10:1–10:21.
54. Forbes MA, Shpilka A (2018) A PSPACE construction of a hitting set for the closure of small algebraic circuits. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing STOC 2018*, ed Servedio RA (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), 1180–1192.
55. Yao AC (1982) Theory and application of trapdoor functions. *23rd Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ), pp 80–91.

56. Arora S, Barak B (2009) *Computational Complexity: A Modern Approach* (Cambridge Univ Press, New York), 1st Ed.
57. Bürgisser P (2013) *Completeness and Reduction in Algebraic Complexity Theory* (Springer Science & Business Media, Springer, Berlin), Vol 7.
58. Kabanets V, Russell I (2004) Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput Complexity* 13:1–46.
59. Clausen M, Dress A, Grabmeier J, Karpinski M (1991) On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theor Computer Sci* 84:151–164.
60. Gupta A, Kamath P, Kayal N, Saptharishi R (2013) Arithmetic circuits: A chasm at depth three. *54th Annual IEEE Symposium on Foundations of Computer Science FOCS* (IEEE, Piscataway, NJ), pp 578–587.
61. Agrawal M, Vinay V (2008) Arithmetic circuits: A chasm at depth four. *49th Annual IEEE Symposium on Foundations of Computer Science FOCS* (IEEE, Piscataway, NJ), pp 67–75.
62. Martin F (2009) Faster integer multiplication. *SIAM J Comput* 39:979–1005.
63. Le Gall F (2014) Powers of tensors and fast matrix multiplication. *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, eds Nabeshima K, Nagasaka K, Winkler F, Szanto A (ACM, New York), pp 296–303.
64. Fischer I (1994) Sums of like powers of multivariate linear forms. *Math Mag* 67:59–61.
65. Agrawal M, Ghosh S, Saxena N (2018) Bootstrapping variables in algebraic circuits. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, eds Diakonikolas I, Kempe D, Henzinger M (ACM, New York), pp 1166–1179.
66. Kumar M, Saptharishi R, Tengse A (2019) Near-optimal bootstrapping of hitting sets for algebraic circuits. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, ed Chan TM (SIAM, Philadelphia), pp 639–646.
67. Saha C, Saptharishi R, Saxena N (2009) The power of depth 2 circuits over algebras. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science FSTTCS*, eds Kannan R, Kumar KN (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), pp 371–382.
68. Forbes MA, Ghosh S, Saxena N (2018) Towards blackbox identity testing of log-variate circuits. *45th International Colloquium on Automata, Languages, and Programming ICALP*, eds Chatzigiannakis I, Kaklamanis C, Marx D, Sannella D (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), pp 54:1–54:16.