Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits*

Zeev Dvir† Amir Shpilka‡ Amir Yehudayoff§

Abstract

In this paper we show that lower bounds for bounded depth arithmetic circuits imply derandomization of polynomial identity testing for bounded depth arithmetic circuits. More formally, if there exists an explicit polynomial f that cannot be computed by a depth d arithmetic circuit of small size then there exists an efficient deterministic black-box algorithm to test whether a given depth $d-5$ circuit that computes a polynomial of relatively small individual degrees is identically zero or not. In particular, if we are guaranteed that the tested circuit computes a multilinear polynomial then we can perform the identity test efficiently. To the best of our knowledge this is the first hardness-randomness tradeoff for bounded depth arithmetic circuits.

The above results are obtained using the arithmetic Nisan-Wigderson generator of Kabanets and Impagliazzo together with a new theorem on bounded depth circuits, which is the main technical contribution of our work. This theorem deals with polynomial equations of the form $P(x_1,\ldots,x_n,y)\equiv 0$ and shows that if P has a circuit of depth d and size s and if the polynomial $f(x_1,\ldots,x_n)$ satisfies $P(x_1,\ldots,x_n,f)\equiv 0$ then f has a circuit of depth $d+3$ and size $\mathrm{poly}(s,m^r)$, where m is the total degree of f and r is the degree of y in P. This circuit for f can be found probabilistically in time $\mathrm{poly}(s,m^r)$.

In the other direction we observe that the methods of Kabanets and Impagliazzo can be used to show that derandomizing identity testing for bounded depth circuits implies lower bounds for the same class of circuits. More formally, if we can derandomize polynomial identity testing for bounded depth circuits then NEXP does not have bounded depth arithmetic circuits. That is, either NEXP $\not\subseteq P/\mathrm{poly}$ or the Permanent is not computable by polynomial size bounded depth arithmetic circuits.

*This paper appeared in a preliminary form in The Proc. of the 40th ACM Symposium on Theory of Computing (STOC 2008).

†Department of Computer Science, Weizmann institute of science, Rehovot, Israel. zeev.dvir@weizmann.ac.il. Research supported by Binational Science Foundation (BSF) grant, by Israel Science Foundation (ISF) grant and by Minerva Foundation grant.

‡Faculty of Computer Science, The Technion, Haifa, Israel. shpilka@cs.technion.ac.il. Research supported by the Israel Science Foundation (grant number 439/06).

§Department of Computer Science, Weizmann institute of science, Rehovot, Israel. amir.yehudayoff@weizmann.ac.il. Research supported by Binational Science Foundation (BSF) grant, by Minerva Foundation grant, by Israel Science Foundation (ISF) grant and by the Israel Ministry of Science (IMOS) – Eshkol Fellowship.

1 Introduction

The role of randomness in computation is a fundamental question in Complexity Theory. Phrased in its most general terms it asks "Do we really need random bits to do that?", where 'that' can be a probabilistic algorithm, interactive protocol, cryptographic application etc. When asking this question in the setting of probabilistic polynomial time algorithms we are actually asking whether BPP=P. In recent years there have been many works giving strong evidence that indeed BPP=P in the form of Hardness-Randomness tradeoffs. These results prove that BPP=P (or some weaker collapse) under the hypothesis that there exist 'explicit' boolean functions that cannot be computed/approximated using small circuits. Since we believe that hard functions exist, we also believe that BPP=P. A partial list of references on this topic includes [NW94, IW97, STV01, ISW01, SU05].

One of the most natural problems in BPP is the problem of Polynomial Identity Testing (PIT) (in fact, PIT belongs to the class coRP $\subseteq$ BPP). In this problem we are given as input a polynomial, represented in some succinct form (say by an arithmetic circuit or formula), and we are asked whether it is the identically zero polynomial[1]. Using the well-known Schwartz-Zippel Lemma [Sch80, Zip79] it is known that evaluating the polynomial at a random point, chosen from a sufficiently large set, is enough in order to determine, with high probability of success, whether the polynomial is identically zero or not. The main question is whether there is an efficient deterministic algorithm for PIT. This problem is considered as one of the central problems in the field of derandomization, partially due to the large number of algorithmic problems that reduce to it (see [AB03, AKS04, Lov79, MVV87, CRS95, LV98]).

In [KI04] Kabanets and Impagliazzo showed that the PIT problem described above can be derandomized if we assume that there exists an explicit polynomial (say the Permanent) that cannot be computed using a small arithmetic circuit. This result is purely arithmetic and does not follow from previous works on boolean derandomization. The two main technical tools used to prove this result are an arithmetic version of the Nisan-Wigderson Generator [NW94] and the polynomial factorization algorithm of Kaltofen [Kal89]. It should be mentioned here that the 'main' result of [KI04] was actually a theorem in the other direction: derandomizing PIT implies circuit lower bounds. This showed that derandomizing PIT might be more difficult then once imagined, since it seems that we are very far from proving explicit circuit lower bounds.

The apparent connection between PIT and circuit lower bounds suggests that it might be beneficial to consider restricted versions of the PIT problem. A natural restriction would be to assume that the input polynomial is given by a 'simpler' representation than a circuit (or even a formula). Since we have lower bounds for restricted models we might be able to solve the PIT versions for the same models. In [GKS90, BOT88, KS01] (and many other papers) deterministic PIT algorithms for depth 2 arithmetic circuits were given. More recently, [RS05] gave a polynomial time PIT algorithm for non-commutative formulas. In another line of work [DS06, KS07b, KS07a, AM07, SS09, KS09] gave PIT algorithms for depth 3 circuits with bounded top fan-in. This should be compared to the best lower bounds for depth 3 circuits which are exponentially large over finite fields [GK98, GR00] and quadratic for characteristic zero fields [SW01]. For depth larger than 3, and fields other than $\mathbb{F}_2$, the only lower bounds are slightly super-linear [BS83, Str73, Pud94, RS05, Raz07].

In this work we consider the problem of PIT for bounded depth arithmetic circuits. Roughly speaking, we show that a circuit lower bound for bounded depth arithmetic circuits would give an efficient deterministic PIT algorithm for bounded depth circuits with a certain limitation on

[1] Note that this is a syntactic requirement. In particular the polynomial $x^2 - x$ is not the zero polynomial although it computes the zero function over $\mathbb{F}_2$.

their degrees (see the Section 1.1 for the precise formulation). This is an *arithmetic* Hardness-Randomness result that further emphasizes the connection between lower bounds and derandomization. One could also hope that in the (near?) future we will succeed in proving lower bounds for bounded depth arithmetic circuits and then, using this work, we would also have unconditional deterministic PIT algorithms for circuits of the same kind.

In order to prove our results we combine the arithmetic Nisan-Wigderson Generator from [KI04] together with a new theorem, Theorem 4, that bounds the bounded depth complexity of a polynomial root $y = f(x_1, \ldots, x_n)$ of an equation $P(x_1, \ldots, x_n, y) \equiv 0$ in terms of the bounded depth complexity of P (in fact, we have a probabilistic algorithm for.finding the bounded depth circuit for f). This theorem replaces the polynomial factorization algorithm of [Kal89] in the proof of [KI04]. We note that the methods of [Kal89] do not seem to work in the bounded depth case since, by the sequential nature of Kaltofen's algorithm, the circuit for the root $f(x_1, \ldots, x_n)$ is always of large depth, even if P has a bounded depth circuit (Kaltofen's algorithm constructs a circuit for $f(x_1, \ldots, x_n)$ using a linear number of steps, each step increasing the depth of the circuit by at least one; hence, it is not clear whether the depth of the circuit can be made small).

1.1 Our results

We proceed by giving the definitions necessary to state our results formally, starting with the formal definitions regarding arithmetic circuits. An *arithmetic circuit* over the field $\mathbb{F}$ and the variables $x_1, \ldots, x_n$ is a directed acyclic graph labelled as follows: Gates of in-degree zero are labelled by either a variable or a field element, and gates with positive in-degree are labelled by either $+$ or $\times$. An edge can be labelled by a field element. An arithmetic circuit computes a polynomial in the obvious way, where a constant on an edge multiplies the polynomial that 'enters' the edge. The *size* of a circuit is the number of edges in it, and the *depth* of a circuit is the length of the longest directed path in it.

We continue by defining three variants of the PIT problem. The problems are ordered from the most general one to the most restricted one. Our results apply only to the last (most restricted) version, but we give all three definitions in order to give a clearer picture.

Problem 1. CPIT($\mathbb{F}$) - Circuit Polynomial Identity Testing Over $\mathbb{F}$ *:*

- *Input: An arithmetic circuit $C(x_1, \ldots, x_n)$ of size poly(n) over a field $\mathbb{F}$.*
- *Output: Does $C(x) \equiv 0$?*

Problem 2. CPIT^d($\mathbb{F}$) - Depth d Circuit PIT Over $\mathbb{F}$ *:*

- *Input: An arithmetic circuit $C(x_1, \ldots, x_n)$ of size poly(n) and depth d over a field $\mathbb{F}$.*
- *Output: Does $C(x) \equiv 0$?*

Problem 3. CPIT^d_r($\mathbb{F}$) - Depth d Circuit PIT Over $\mathbb{F}$ for polynomials with individual degrees at most r*:*

- *Input: An arithmetic circuit $C(x_1, \ldots, x_n)$ of size poly(n) and depth d over a field $\mathbb{F}$ computing a polynomial with individual degrees at most r (possibly the zero polynomial).*
- *Output: Does $C(x) \equiv 0$?*

We note that each of the problems can be considered either in the non black-box setting and in the black-box setting. In the non black-box setting we are given the arithmetic circuit and we can use its graph of computation in order to decide whether it computes the zero polynomial or not. In the black-box model we do not have the circuit at our disposal and our only access to the polynomial computed by the circuit is via queries. In particular, a black-box PIT algorithm is no other than a *hitting set*. Namely, a set of points such that if the polynomial computed by the circuit is non-zero then it does not vanish when evaluated on the points of the hitting set. All the PIT algorithms that we give are in the black-box model.

When considering the uniform complexity of the above problems we restrict our attention to finite fields and to the field of rational numbers $\mathbb{Q}$. Elements in these fields can be represented using finite bit strings, and so the standard Turing machine model is sufficient to describe algorithms over these fields. This will save a bit on unimportant technicalities. When working over a finite field $\mathbb{F}_{p^r}$, p prime, we will assume (as is done in [KI04]) that we have at our disposal an irreducible polynomial of degree r over $\mathbb{F}_p$.

Remark 1.1 (Syntactic vs. semantic restrictions)**.** *The definition of $CPIT_r^d(\mathbb{F})$ does not contain any* syntactic *restrictions on the circuit C. In particular, C can have intermediate gates computing polynomials that have individual degrees larger than r, as long as the output has individual degrees at most r. This can be thought of as a* semantic *restriction - applying only to the output of the computation.*

Our PIT algorithms will assume that there exists some 'explicit' polynomial that is hard for small circuits of bounded depth. To simplify some of the proofs we will assume that this polynomial is also multilinear. We note that this requirement is not really needed since if we had a hard polynomial which was not multilinear, but had, say, polynomial degree in each variable, we could easily derive from it an explicit hard *multilinear* polynomial with only a polynomial deterioration in the hardness parameters. More precisely, replacing x_i^r, for $r > 1$ with $y_{i0}^{r_0} \cdot \ldots \cdot y_{is}^{r_s}$, where $(r_0 \ldots r_s)$ is the binary representation of r, gives a new multilinear polynomial in a slightly larger number of variables. This polynomial is at least as hard as the original polynomial which can be recovered from it by the substitution $y_{ij} = x_i^{2^j}$.

The following two definitions capture the notions of 'explicitness' and 'hardness' necessary to formulate our results.

Definition 1.2 (Explicit polynomial)**.** *Let $\mathbb{F}$ be a finite field or the field of rational numbers. Let $\tilde{p} = \{p_m\}_{m=1}^{\infty}$ be a sequence of multilinear polynomials such that $p_m \in \mathbb{F}[x_1, \ldots, x_m]$ for each m. We say that the sequence $\tilde{p}$ is* explicit *if*

1. *All the coefficients of p_m have size (in bits) polynomial in m.*
2. *There exists a Turing machine M that on input m runs in time $2^{O(m)}$ and outputs a list (of length 2^m) of all the coefficients of $p_m(x_1, \ldots, x_m)$.*

Definition 1.3 (Hardness against circuits)**.** *Let $\mathbb{F}$ be a field and let $\mathbb{E}$ be an extension field of $\mathbb{F}$. Let $p \in \mathbb{F}[x_1, \ldots, x_m]$ be some polynomial. We say that p is $(s, d, \mathbb{E})$-hard if p cannot be computed by an arithmetic circuit of size s and depth d over the extension field $\mathbb{E}$.*

The next two theorems state our main result when the underlying field is $\mathbb{Q}$ (Theorem 1) and over finite fields (Theorem 2).

Theorem 1. *Let d be an integer, $\epsilon > 0$ a real number. Suppose there exists an explicit sequence $\tilde{p} = \{p_m\}_{m=1}^{\infty}$ of multilinear polynomials such that for each m the polynomial p_m belongs to $\mathbb{Q}[x_1, \ldots, x_m]$*

and is $(2^{m^\epsilon}, d, \mathbb{Q})$*-hard. Then the problem* $CPIT^{d'}_{polylog(n)}(\mathbb{Q})$ *can be solved deterministically (in the black-box model) in time* $n^{polylog(n)}$*, where* $d' = d - 5$

Theorem 2. *Let* d *be an integer,* $\epsilon > 0$ *a real number,* $\mathbb{F}$ *a finite field. Suppose there exists an explicit sequence* $\tilde{p} = \{p_m\}_{m=1}^{\infty}$ *of multilinear polynomials such that for each* m *the polynomial* p_m *belongs to* $\mathbb{F}[x_1, \ldots, x_m]$ *and is* $(2^{m^\epsilon}, d, \mathbb{E})$*-hard for some finite extension* $\mathbb{E}$ *of* $\mathbb{F}$ *satisfying* $|\mathbb{E}| > 2^m$*. Then the problem* $CPIT^{d'}_{polylog(n)}(\mathbb{F})$ *can be solved deterministically (in the black-box model) in time* $n^{polylog(n)}$*, where* $d' = d - 5$*.*

Remark 1.4 (Higher individual degrees)**.** *We note that if we are satisfied with having sub-exponential time deterministic PIT algorithms (this would be a weaker, but still highly interesting, derandomization result) then we can replace* $CPIT^{d'}_{polylog(n)}(\mathbb{F})$ *with* $CPIT^{d'}_{n^{o(1)}}(\mathbb{F})$ *in the above theorems. This follows by making a small modification to the proof, in a similar fashion to [KI04] (we leave the details to the interested reader).*

We observe that a converse to Theorem 1 and Theorem 2 is also true. Namely, that derandomizing identity testing for bounded depth circuits implies circuit lower bounds for the same model.

Theorem 3. *The following three assumptions cannot be simultaneously true.*

1. NEXP $\subseteq$ P/poly*,*
2. *Permanent is computable by polynomial size* depth d *arithmetic circuits over* $\mathbb{Q}$*,*
3. $\mathrm{CPIT}^d(\mathbb{Q})$ *is in* NSUBEXP*.*

We stress that Theorem 3 above is a simple observation, since the same theorem for general circuits (without depth restriction) appears in [KI04] and the proof requires almost no modifications to hold also in the bounded depth model.

A key ingredient of our proof is the following theorem, which relates the bounded depth complexity of a polynomial $y = f(x_1, \ldots, x_n)$ that satisfies $P(x_1, \ldots, x_n, y) = 0$ to that of $P(x, y)$. The theorem tells us that we can find a bounded depth circuit for f efficiently (using randomness). The algorithmic part of the theorem is of no importance for the hardness-randomness tradeoffs – for the tradeoffs we just need the *existence* of a small bounded depth circuit for f.

Theorem 4. *Let* n, s, r, m, t, d *be integers such that* $s \geq n$*. Let* $\mathbb{F}$ *be a field which has at least* $2mt$ *elements. Let* $P(x_1, \ldots, x_n, y) \in \mathbb{F}[x_1, \ldots, x_n, y]$ *be a non-zero polynomial with* $\deg(P) \leq t$ *and* $\deg_y(P) \leq r$ *such that* P *has an arithmetic circuit of size* s *and depth* d *over* $\mathbb{F}$*. Let* $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ *be a polynomial with* $\deg(f) = m$ *such that* $P(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) \equiv 0$*. Then, there exists a probabilistic algorithm that is given as input the circuit for* P*, the numbers* r, m, t*, and oracle access to* f*, runs in time* $poly(s, m^r)$*, and outputs a circuit of depth*[2] $d + 3$ *computing* f*, with probability at least* $2/3$*.*

We now discuss shortly the connection between Theorem 4 and Kaltofen's algorithm for factoring polynomials. The proof of Theorem 4 actually shows that given a small constant depth circuit for P, there is a probabilistic algorithm for finding small constant depth circuits for all the polynomials f that are roots of P (see the discussion after the proof of Lemma 3.1 for more

[2]The bound $d + 3$ on the depth can be improved to $d + 2$ if we assume that the circuit for P has a multiplication gate as its top most (output) gate. It can also be improved to $d + 2$ if the layer above the input gates is of $+$ gates. If both conditions hold then the depth is $d + 1$.

details). Kaltofen's factorization algorithm can also be used to output small circuits for the roots of P as well; indeed, if f is a root of P, then $(y - f)$ is a factor of P. Thus, Kaltofen's algorithm solves a more general question than the root finding problem. However, in Kaltofen's algorithm we have no guarantee on the depth of the circuits for the different factors whereas Theorem 4 gives a good bound on the depth of each root. This is the main advantage of Theorem 4 over Kaltofen's algorithm. On the other hand, a drawback of Theorem 4 is the exponential dependency on r; the bounded depth circuit we obtain for f is small only if r is small, for example, it is of quasi-polynomial size only for poly-logarithmic r. This blowup does not occur in Kaltofen's algorithm; in his algorithm the circuit for f is always polynomial. The main reason for this blowup is our insistence on constant depth; if we did not care about the depth, we would not have this blowup (see the proof of Lemma 3.1). We note that this blowup is the reason for the restriction on the individual degrees in Theorems 1 and 2.

1.2 A discussion of recent results

After this paper was sent to review, Agrawal and Vinay proved that in order to derandomize the PIT problem for general arithmetic circuits, it is enough to derandomize the problem for depth 4 circuits [AV08]. Namely, a polynomial size hitting set for polynomial size depth 4 circuits implies the existence of a polynomial, that can be computed in PSPACE, whose arithmetic circuit complexity is exponential. Applying the generator construction of [KI04] on this polynomial we get an $n^{O(\log n)}$ time PIT algorithm for general arithmetic circuits. In addition, [AV08] show that a lower bound of the form $\exp(\Omega(n))$ for depth 4 circuits implies a similar result for general circuits. However, we note that if we have a lower bound of the form $\exp(n^{\epsilon})$ for depth d circuits then the Agrawal-Vinay result does not yield any identity testing algorithm for bounded depth circuits nor a lower bound for general arithmetic circuits, whereas our result does.

1.3 Organization

In Section 2 we give notations and preliminary claims that will be used in later sections. Section 3 contains our main technical contribution: a 'root finding' theorem for bounded depth circuits. Finally, in Section 4 we use this theorem to prove Theorems 1 and 2. A sketch of the proof of Theorem 3 is given in Section 5.

2 Preliminaries

2.1 Notations

We denote $[n] = \{1, \ldots, n\}$. Let $p \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. We write $\deg(p)$ for the total degree of p and $\deg_{x_i}(p)$ for the individual degree of p in the variable x_i. We sometimes write $p(x)$ to denote $p(x_1, \ldots, x_n)$. In the same way, we sometimes denote a polynomial $p \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_s]$ by $p(x, y)$. Let $S \subset [n]$ be a set of size $m > 0$ and write $S = \{i_1, \ldots, i_m\}$ where $i_1 < \ldots < i_m$. Given a polynomial $q \in \mathbb{F}[y_1, \ldots, y_m]$ and $x = (x_1, \ldots, x_n)$ we denote by $q(x|_S)$ the restriction of q to the variables in S, namely $q(x|_S) \triangleq q(x_{i_1}, \ldots, x_{i_m})$. For a polynomial $f(x) \in \mathbb{F}[x_1, \ldots, x_n]$ we denote by $H_i[f]$ the homogenous part of degree i of f and $H_{\leq i}[f] = \sum_{j \leq i} H_j[f]$. Namely, $H_i[f]$ is the sum of all terms of degree exactly i in f.

2.2 Combinatorial designs

We quote the standard result on the combinatorial designs of Nisan and Wigderson.

Lemma 2.1. *[NW94] Let n, m be integers such that $n < 2^m$. There exists a family of sets $S_1, \ldots, S_n \subset [l]$ such that*

- *$l = O(m^2/\log(n))$.*
- *For each $i \in [n]$, $|S_i| = m$.*
- *For every $1 \leq i < j \leq n$, $|S_i \cap S_j| \leq \log(n)$.*

Moreover, this family of sets can be computed deterministically in time poly$(n, 2^l)$.

2.3 The Schwartz-Zippel Lemma

The following generalization of the fundamental theorem of algebra is due to Schwartz and Zippel [Sch80, Zip79].

Lemma 2.2 (Schwartz-Zippel)**.** *Let $\mathbb{F}$ be a field and let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non zero polynomial with degree at most r. Then, for any finite subset $S \subset \mathbb{F}$ we have*

$$|\{c \in S^n : f(c) = 0\}| \leq r \cdot |S|^{n-1}.$$

The Schwartz-Zippel Lemma gives a trivial (but not very efficient) deterministic PIT algorithm for circuits. We call this algorithm the 'brute force' algorithm.

Algorithm 2.3. *[Brute Force PIT] Given an arithmetic circuit $C(x_1, \ldots, x_n)$ over $\mathbb{F}$ and a bound r on its degree we test whether $C \equiv 0$ as follows. We pick a set $S \subset \mathbb{F}$ of size $r+1$ (if $\mathbb{F}$ is smaller than $r+1$ we allow S to be in some extension field). We then go over all assignments $a \in S^n$ and check whether $C(a) = 0$. If all the tests returned zero then we say that $C \equiv 0$, otherwise we say that $C \not\equiv 0$.*

2.4 Computing the homogenous parts of a circuit

The next lemma says that if f has a circuit of small size and depth then so do the polynomials $H_i[f]$ (assuming the field is not too small).

Lemma 2.4. *Let $\mathbb{F}$ be a field which has at least $m+1$ distinct elements. Let $f(x) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial with $\deg(f) = m$ such that f has a circuit of size s and depth d. Then there exists a circuit of size $s' = O(s \cdot m)$ and depth $d' = d+1$ with $m+1$ outputs computing $H_0[f], \ldots, H_m[f]$. Moreover, if the top most gate in the circuit for f is an addition gate then the we have $d' = d$.*

Proof. Let z be a new formal variable and define

$$g(x_1, \ldots, x_n, z) = f(x_1 \cdot z, \ldots, x_n \cdot z). \tag{1}$$

Notice that

$$g(x, z) = \sum_{i=0}^{m} H_i[f] \cdot z^i. \tag{2}$$

Computing all of the $H_i[f]$ is now done by treating g as a univariate polynomial in z and recovering its 'coefficients' using evaluations on a large enough set of points. More formally, let $c_0, \ldots, c_m \in \mathbb{F}$

be $m+1$ distinct elements. Let Γ be an $(m+1)\times(m+1)$ matrix whose i'th row is $(1, c_i, c_i^2, \ldots, c_i^m)$. Let α be the column vector $(H_0[f], H_1[f], \ldots, H_m[f])^t$ and $\beta = (g(x, c_0), g(x, c_1), \ldots, g(x, c_m))^t$. Using (2) we see that $\beta = \Gamma \cdot \alpha$ and so, since Γ is invertible, we have $\alpha = \Gamma^{-1} \cdot \beta$. Computing all the entries of β (in parallel) can be done in depth d and size $O(s \cdot m)$ using the identity in (1). Now, computing the entries of α can be done by adding another layer of addition gates computing the linear transformation Γ^{-1}. This increases the depth by one unless the top most gate is already an addition gate. □

3 Roots of equations with polynomial coefficients

We now prove Theorem 4 – we show that a solution $y = f(x_1, \ldots, x_n)$ of a polynomial equation $P(x_1, \ldots, x_n, y) = 0$ cannot have bounded depth complexity significantly larger than that of $P(x, y)$, when the degree of the variable y in P is not too large.

The heart of the proof of Theorem 4 is the following lemma that shows that (under certain conditions on the derivative of P) we can 'approximate' the polynomial root $y = f(x)$ of the equation $P(x, y) = 0$ using a polynomial in the coefficients of P. Each approximation gives the monomials of $f(x)$ up to some degree k plus some other monomials of higher degree than k (this is the 'error' term). In the proof of Theorem 4 we will use only the last approximation (when $k = m$) to construct a circuit for $f(x)$.

Lemma 3.1. *Let $\mathbb{F}$ be a field, let $P \in \mathbb{F}[x_1, \ldots, x_n, y]$ be such that $\deg_y(P) = r$ and let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be such that $P(x, f(x)) \equiv 0$ and $\frac{\partial P}{\partial y}(0, f(0)) = \xi_0 \neq 0$. Write $P(x, y) = \sum_{i=0}^{r} C_i(x) \cdot y^i$. Then, for each $k \geq 0$, there exists a polynomial $Q_k \in \mathbb{F}[z_0, \ldots, z_r]$ of degree at most k such that*

$$H_{\leq k}[f(x)] \equiv H_{\leq k}[Q_k(C_0(x), \ldots, C_r(x))].$$

Moreover, given $f(0)$ and a size-s circuit for P we can construct, in time poly(s, r, k), a circuit for Q_k.

Although we work over finite fields as well, we use the usual definition of a partial derivative as over $\mathbb{R}, \mathbb{C}$. We defer the proof of Lemma 3.1 to section 3.2 and proceed to give the proof of Theorem 4.

3.1 Proof of Theorem 4

We start with a simple claim regarding the complexity of computing partial derivatives.

Claim 3.2. *For every $1 \leq j \leq r$, the partial derivative $\frac{\partial^j P}{\partial y^j}(x, y)$ has an arithmetic circuit of size poly(s, r) and depth at most $d + 1$ (the depth remains d if the top gate is an addition gate). This circuit can be found in time poly(s, r).*

Proof. Write $P(x, y) = \sum_{j=0}^{r} C_j(x) \cdot y^j$. Using the same trick as in the proof of Lemma 2.4, we get that there exists a circuit D of size poly(s, r) and depth $d + 1$ with $r + 1$ outputs computing $C_0(x), \ldots, C_r(x)$. To construct D we need to evaluate C in $r + 1$ values and interpolate, that is, invert a $(r + 1) \times (r + 1)$ matrix. This can be done in time poly(r).

Now, each partial derivative is a linear combination of products $C_j(x) \cdot y^i$. We can compute each term of the form $C_j(x) \cdot y^i$ using a circuit of size poly(s, r) and depth $d+1$ by multiplying each multiplication gate at the second highest level of the circuit by y^j (recall that the top most gate is an addition gate now). The final circuit for the derivative will be composed of a linear combination

of these circuits (which have a top most addition gate) and so will also have depth $d+1$. Notice that if the top most gate in the original circuit was an addition gate then the resulting circuit will have depth d. Given D and j, the time it takes to construct a circuit computing $\frac{\partial^j P}{\partial y^j}$ is also $\text{poly}(s, r)$. □

Part I - Preparations: We first note that we can assume w.l.o.g. that P is computed by a depth d circuit with an addition gate at the top. Otherwise, if $P = P_1 \cdot P_2 \cdots P_r$, where each P_i has a size at most s and depth $d-1$ circuit with an addition gate at the top, then for some i we have that $P_i(x, f(x)) \equiv 0$. To find P_i, we can use the Schwartz-Zippel Lemma (Lemma 2.2). Namely, we choose c uniformly at random from S^n, where S is a subset of $\mathbb{F}$ of size at least $2mt$. The probability that $P_j(c, f(c)) = 0$, when $P_j(x, f(x)) \not\equiv 0$, is at most $1/2$, as $\deg(P_j(x, f(x))) \leq mt$. By repetition, this probability can be made arbitrary small, e.g., smaller than r^{-2}. Thus, we can find P_i, with probability at least $5/6$, in time $\text{poly}(r, s)$ (we use the oracle access to f).

Observing Claim 3.2 we can assume w.l.o.g. that

$$\frac{\partial P}{\partial y}(x, f(x)) \not\equiv 0. \tag{3}$$

Otherwise we could replace P with the first $\tilde{P} = \frac{\partial^j P}{\partial y^j}(x, y)$ such that $\tilde{P}(x, f(x)) = 0$ and $\frac{\partial \tilde{P}}{\partial y}(x, f(x)) \not\equiv 0$. The loss in the size incurred by Claim 3.2 is too small to be a problem. As we assume that the top gate is an addition gate we get that the depth remains the same. We can find $\tilde{P}$ using randomness and the Schwartz-Zippel Lemma again, with probability at least $5/6$, in time $\text{poly}(r, s)$. This is done iteratively: for every j from 1 to r, check if $\frac{\partial^j P}{\partial y^j}(x, f)$ is zero, if it is zero continue to $j+1$, and if it is not zero stop. In particular, we found a point $x^0 \in \mathbb{F}^n$ such that $\frac{\partial P}{\partial y}(x^0, f(x^0)) \neq 0$.

We can further assume w.l.o.g. that $x^0 = 0$. Otherwise we could prove the theorem for the polynomials $P(x + x^0, y)$ and $f(x + x^0)$, and translate the result back to the original P and f (one can verify that the earlier condition, given by (3), on the derivative not being identically zero is maintained by this transformation). The depth of the circuit may increase by 1 by this translation.

To conclude, with probability at least $2/3$, we are now in a situation where

$$\frac{\partial P}{\partial y}(0, f(0)) \neq 0.$$

Part II - Using Lemma 3.1: Write

$$P(x, y) = \sum_{j=0}^{r} C_j(x) \cdot y^j.$$

Applying Lemma 3.1 (proved in the next subsection) with $k = \deg(f) = m$, we get that there exists a polynomial $Q \in \mathbb{F}[z_0, \ldots, z_r]$ of degree at most m such that

$$f(x) \equiv H_{\leq m}[f(x)] \equiv H_{\leq m}[Q(C_0(x), \ldots, C_r(x))].$$

In fact, Lemma 3.1 guarantees that we can find a circuit computing Q in time $\text{poly}(r, s, m)$ (we can use the oracle access to f). As $\deg(Q) \leq m$, we can use interpolation to find $\{Q_\alpha\}_{\alpha \in I_m}$ so that

$$Q(z_0, \ldots, z_r) = \sum_{\alpha \in I_m} Q_\alpha \cdot \prod_{i=0}^{r} z_i^{\alpha_i},$$

where $I_m \triangleq \{(\alpha_0, \ldots, \alpha_r) \in \mathbb{N}^{r+1} \mid \sum_i \alpha_i \leq m\}$. As Q has at most $(m+1)^{r+1}$ monomials, this requires time $\text{poly}(s, m^r)$.

Part III - Finding a circuit for f(x): Observing the proof of Claim 3.2, we see that we can find depth d circuits for the polynomials $C_0(x), \ldots, C_r(x)$ in time $\text{poly}(s, r)$ (as the top gate is $+$). In Part II, we found a depth two circuit of size $\text{poly}(m^r)$ computing Q. Therefore, we can find in time $\text{poly}(s, m^r)$ a circuit of depth $d+2$ computing the polynomial $g(x) \triangleq Q(C_0(x), \ldots, C_r(x))$; compose the circuit for Q with the circuits for $C_0, \ldots, C_r$. Finally, Lemma 2.4 tells us that $f(x) = H_{\leq k}[g(x)]$ can be computed by a circuit of size $\text{poly}(s, m^r)$ and depth $d+2$ (we use the fact that $|\mathbb{F}| > t \cdot m \geq \deg(g)$, and that the depth two circuit for Q has a $+$ gate in the top). Again, this takes time $\text{poly}(s, m^r)$. As we need to translate the input by x^0 (recall part I), the circuit for f may be of depth $d+3$ (if the layer above the inputs is of $+$ gates, then the circuit for f has depth $d+2$). □

3.2 Proof of Lemma 3.1

We, first, observe that it is enough to prove the lemma *without any restriction* on the degree of Q_k, as we now explain. Denote $\tilde{C}_i(x) = C_i(x) - C_i(0)$. Every monomial in $\tilde{C}_i$ has degree at least one. We, in fact, describe how to, in time $\text{poly}(s, r, k)$, construct a circuit for a polynomial $\tilde{Q}_k(z_0, \ldots, z_r)$ so that $H_{\leq k}[f] = H_{\leq k}\left[\tilde{Q}_k(\tilde{C}_0, \ldots, \tilde{C}_r)\right]$. Given such a circuit for $\tilde{Q}_k$, we can, in time $\text{poly}(s, r, k)$, compute a circuit for $\hat{Q}_k = H_{\leq k}[\tilde{Q}_k]$, using the methods described in [Str73] and [SY10, Theorem 2.2]. Note that $H_{\leq k}[\tilde{Q}_k]$ is with respect to the variables $z_0, \ldots, z_r$, whereas $H_{\leq k}[f]$, for example, is with respect to x. Since all the polynomials $\tilde{C}_0, \ldots, \tilde{C}_r$ do not have a constant term,

$$H_{\leq k}[f] = H_{\leq k}\left[\hat{Q}_k(\tilde{C}_0, \ldots, \tilde{C}_r)\right].$$

By Claim 3.2, given a circuit of size s for P, we can recover $C_0(0), \ldots, C_r(0)$ in time $\text{poly}(s, r)$. This implies the lemma with

$$Q_k(z_0, \ldots, z_r) = \hat{Q}_k(z_0 - C_0(0), \ldots, z_r - C_r(0)).$$

The degree of Q_k is at most k since it is a translate of a polynomial of degree at most k.

Not having to worry about the degree of $\tilde{Q}_k$ growing, we show how to efficiently construct a circuit for $\tilde{Q}_k$ given a circuit for $\tilde{Q}_{k-1}$. To start this process, we need to address the case $k = 0$. For $k = 0$, we just use[3] the given information $\tilde{Q}_0 = f(0)$.

Next, assume that we constructed a circuit for $\tilde{Q}_{k-1}$. Denote

$$g = \tilde{Q}_{k-1}(\tilde{C}_0, \ldots, \tilde{C}_r).$$

and denote $\xi_0 = P'(0, f(0)) \neq 0$, where $P' = \frac{\partial P}{\partial y}$. We can calculate ξ_0 in time $\text{poly}(s)$. Since the constant term is the same in f and in g, we have $\xi_0 = P'(0, g(0))$. Let $P_0(y)$ be so that $P(x, y) = \sum_{i=0}^{r} \tilde{C}_i(x) \cdot y^i + P_0(y)$. A circuit for $P_0(y)$ can be computed efficiently by $P_0(y) = \sum_{i=0}^{r} C_i(0) y^i$. Set

$$\begin{aligned} &\tilde{Q}_k(z_0, \ldots, z_r) \\ &\quad \triangleq \tilde{Q}_{k-1}(z_0, \ldots, z_r) - (1/\xi_0)\left(\sum_{i=0}^{r} z_i \cdot \tilde{Q}_{k-1}(z_0, \ldots, z_r)^i + P_0(\tilde{Q}_{k-1}(z_0, \ldots, z_r))\right). \end{aligned} \tag{4}$$

[3] Alternatively, this can be done by factoring the polynomial $P(0, y)$. In this way we get several possible values for $f(0)$. We work with each of these values separately. It is interesting that if P has several different roots then they all get different values at the point 0, we discus this in more detail after the proof.

The size of the circuit thus defined for $\tilde{Q}_k$ is larger than the size of the circuit for $\tilde{Q}_{k-1}$ by at most an *additive* factor of $\mathrm{poly}(s,r)$. Overall the circuit-size of $\tilde{Q}_k$ is $\mathrm{poly}(s,r,k)$. By choice,

$$\tilde{Q}_k(\tilde{C}_0(x),\ldots,\tilde{C}_r(x)) = g(x) - (1/\xi_0)P(x,g(x)). \tag{5}$$

We will finish the proof by showing that

$$H_{\leq k}\left[g(x) - (1/\xi_0)\cdot P(x,g(x))\right] = H_{\leq k}\left[f(x)\right]. \tag{6}$$

The following chain of polynomial identities is derived by throwing away (or modifying) terms that have total degree larger than k and using the above information of f and g; in particular, the fact that $H_{\leq k-1}[f(x)] = H_{\leq k-1}[g(x)]$. Let us denote $f_k(x) = H_k[f(x)]$ and similarly for g. Notice that we do not assume any bound on the degree of g.

$$\begin{aligned}
0 &\equiv H_{\leq k}\left[P(x,f(x))\right] \\
&\equiv H_{\leq k}\left[P(x,g(x)+\{f_k(x)-g_k(x)\})\right] \\
&\equiv H_{\leq k}\left[\sum_{i=0}^{r} C_i(x)\cdot(g(x)+\{f_k(x)-g_k(x)\})^i\right] \\
&\equiv H_{\leq k}\left[\sum_{i=0}^{r} C_i(x)\cdot(g(x)^i + i\cdot\{f_k(x)-g_k(x)\}\cdot g(x)^{i-1})\right] \\
&\equiv H_{\leq k}\left[P(x,g(x))\right] + H_{\leq k}\left[\{f_k(x)-g_k(x)\}\cdot P'(x,g(x))\right]
\end{aligned}$$

The polynomial $\{f_k(x)-g_k(x)\}$ contains only monomials of degree at least k (if $f_k(x)\equiv g_k(x)$ then there is nothing to prove). So, in order to get the homogenous part of degree k in $\{f_k(x)-g_k(x)\}\cdot P'(x,g(x))$, we have to take the homogenous part of degree 0 in $P'(x,g(x))$, which is given by $P'(0,g(0)) = \xi_0 \neq 0$, and multiply it by $\{f_k(x)-g_k(x)\}$. We can therefore continue the above chain of identities as follows

$$\begin{aligned}
\ldots &\equiv H_{\leq k}\left[P(x,g(x))\right] + \xi_0\cdot\{f_k(x)-g_k(x)\} \\
&\equiv H_{\leq k}\left[P(x,g(x))\right] + \xi_0\cdot\{H_{\leq k}[f(x)] - H_{\leq k-1}[f(x)] - H_{\leq k}[g(x)] + H_{\leq k-1}[g(x)]\} \\
&\equiv H_{\leq k}\left[P(x,g(x))\right] + \xi_0\cdot\{H_{\leq k}[f(x)] - H_{\leq k}[g(x)]\}.
\end{aligned}$$

Rearranging we get

$$\begin{aligned}
H_{\leq k}[f(x)] &\equiv H_{\leq k}[g(x)] - H_{\leq k}\left[(1/\xi_0)\cdot P(x,g(x))\right] \\
&\equiv H_{\leq k}\left[g(x) - (1/\xi_0)\cdot P(x,g(x))\right].
\end{aligned}$$

□

We now address a subtle issue in the proof of Lemma 3.1. Given a polynomial P, it may have many different roots, say, $f_1,\ldots,f_t$. For every f_i, Lemma 3.1 defines an approximation Q_i, which depends only on a single number, namely $f_i(0)$. It is, perhaps, surprising that given P this number entirely describes f_i. This property follows from the assumption on P'. For example, assume that $P = \prod_{i=1}^{t}(y-f_i)$, and so $P' = \sum_{i=1}^{t}\prod_{j\neq i}(y-f_j)$. Thus, if two different polynomials f_i and f_j admit $f_i(0) = f_j(0)$, then $P'(0,f_i(0)) = P'(0,f_j(0)) = 0$. In this case, the lemma does not guarantee anything.

4 Identity testing using a hard polynomial

In this section we prove Theorem 1 and Theorem 2. Since the proofs are mostly identical we will do them together (making sure to note all the places that are different). For the rest of this section $\mathbb{F}$ will denote either a finite field or the field of rational numbers.

4.1 The main lemma

Lemma 4.1. *Let n, r, s, d be integers and let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non zero polynomial with individual degrees at most r that is computed by a size $s \geq n$ circuit of depth d. Let $m \leq (\log(n))^{O(1)}$ be an integer and let $S_1, \ldots, S_n \subset [l]$ be given by Lemma 2.1 so that $l = (\log(n))^{O(1)}$, $|S_i| = m$ and $|S_i \cap S_j| \leq \log(n)$. Let $p \in \mathbb{F}[z_1, \ldots, z_m]$ be a multilinear polynomial such that*

$$F(y) = F(y_1, \ldots, y_l) \triangleq f\left(p(y|_{S_1}), \ldots, p(y|_{S_n})\right) \equiv 0.$$

Then $p(z)$ can be computed by a circuit of size $\leq (s \cdot m^r)^a$ and depth $d+5$ over $\mathbb{F}$, where a is some absolute constant. If $\mathbb{F}$ is finite then we also require that $|\mathbb{F}| \geq n^2$.

Proof. First notice that we can assume w.l.o.g. that $r < m$, for otherwise the bound on the circuit size for p becomes trivial, even for a depth 2 circuit. We start by defining the 'hybrid' polynomials:

$$\begin{aligned} F_0(x, y) &= f(x_1, \ldots, x_n), \\ F_1(x, y) &= f(p(y|_{S_1}), x_2, \ldots, x_n), \\ &\vdots \\ F_n(x, y) &= f(p(y|_{S_1}), \ldots, p(y|_{S_n})). \end{aligned}$$

By our assumptions we have that $F_0(x, y) \not\equiv 0$ and $F_n(x, y) \equiv 0$. Therefore, there exists an index $0 \leq i < n$ such that

$$F_i(x, y) \not\equiv 0 \quad \text{and} \quad F_{i+1}(x, y) \equiv 0. \tag{7}$$

We would like to fix all the variables $x_{i+2}, \ldots, x_n$ (if $i < n-1$) and the variables in $\{y_j | j \notin S_{i+1}\}$ to values in $\mathbb{F}$ such that the property given by (7) still holds for the restricted versions of F_i and F_{i+1}. This is possible by Lemma 2.2 and using the bound $|\mathbb{F}| \geq n^2 > nrm \geq \deg(F_i)$ if $\mathbb{F}$ is finite. Fixing the aforementioned variables leaves us with equations

$$\begin{aligned} &\tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \ldots, q_i(y|_{S_i \cap S_{i+1}}), x_{i+1}) \not\equiv 0 \\ &\tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \ldots, q_i(y|_{S_i \cap S_{i+1}}), p(y|_{S_{i+1}})) \equiv 0, \end{aligned} \tag{8}$$

where $\tilde{f}$ is simply f with the variables $x_{i+2}, \ldots, x_n$ fixed to some values and $q_j(y|_{S_j \cap S_{i+1}})$ are the polynomials $p(y|_{S_j})$ after we fix the variables $\{y_j | j \notin S_{i+1}\}$. In order to simplify the notations for the rest of the proof we rename our variables and rewrite (8) as follows:

$$\begin{aligned} &g(z_1, \ldots, z_m, w) \not\equiv 0 \\ &g(z_1, \ldots, z_m, p(z_1, \ldots, z_m)) \equiv 0, \end{aligned} \tag{9}$$

where the polynomial $g(z_1, \ldots, z_m, w)$ is obtained by taking the expression

$$\tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \ldots, q_i(y|_{S_i \cap S_{i+1}}), x_{i+1}) \tag{10}$$

and replacing x_{i+1} with w and the variables $\{y_j | j \in S_{i+1}\}$ with $\{z_1, \ldots, z_m\}$ (maintaining their relative order).

Our final step is to apply Theorem 4 on the polynomials $g(z, w)$ and $p(z)$ in order to show that $p(z)$ has a small circuit. Before doing so we will need to bound the size/depth of the circuit computing $g(z, w)$.

Claim 4.2. *The polynomial $g(z, w)$ can be computed by a circuit of size poly(n, s) = poly(s) and depth $d + 2$.*

Proof. Since $g(z, w)$ has the same circuit complexity as the expression in (10) we will find a circuit for that expression. The polynomials $q_j(y|_{S_j \cap S_{i+1}})$ are multilinear polynomials of at most $\log(n)$ variables and as such can be computed by a depth two circuit of size $\leq n$. Plugging these circuits into the circuit for f (or actually to the restricted circuit for $\tilde{f}$), which is of size s and depth d, gives the required circuit for $g(z, w)$ (after renaming the variables). □

Notice that $\deg_w(g) = \deg_{x_{i+1}}(f) \leq r$. We can therefore use Theorem 4 to get that $p(z)$ has a circuit of depth $d + 5$ and size $\leq (s \cdot m^r)^a$. If $\mathbb{F}$ is finite then the conditions of Theorem 4 are satisfied since $|\mathbb{F}| \geq n^2 > \max\{r + 1, \deg(g) \cdot \deg(p) + 1\}$. □

4.2 Proof of Theorems 1 and 2

The proof is divided into three parts: a description of the PIT algorithm, a proof of its correctness and an analysis of its running time. Recall that in Theorem 2 we have an underlying finite field $\mathbb{F}$ and a polynomial $p \in \mathbb{F}[z_1, \ldots, z_m]$ which is hard even for circuits over some finite extension $\mathbb{E}$ of $\mathbb{F}$ of size at least 2^m. In order to simplify the notations we will assume for the remainder of this section that $\mathbb{F}$ *itself* has size at least 2^m. This will still imply the theorem since the question whether a circuit is identically zero or not is unchanged when we work over an extension field. Notice also that since in our proof $m = \text{polylog}(n)$ the size of $\mathbb{F}$ is quasi-polynomial in n – the number of inputs in the input polynomial.

The Hitting set: Given a size bound $s \leq n^c$ (for some constant c) a depth bound d' and a bound $r = \text{polylog}(n)$ on the maximal degrees, we fix A to be some large constant to be determined later (in the analysis) and let $m = \lceil (A \cdot r \cdot \log(n))^{3/\epsilon} \rceil = \log(n)^{O(1)}$ (recall that ϵ is such that p_m is $(2^{m^\epsilon}, d, \mathbb{F})$-hard). The hitting set is constructed as follows:

1. Construct a design $S_1, \ldots, S_n \subset [l]$ as in Lemma 2.1 such that $l = O(m^2/\log(n)) = \log(n)^{O(1)}$, $|S_i| = m$ and $|S_i \cap S_j| \leq \log(n)$.
2. Pick a subset $T \subset \mathbb{F}$ of size $n \cdot m \cdot r + 1$. Construct the set of all n-tuples of the form $(p_m(y|_{S_1}), \ldots, p_m(y|_{S_n}))$ by going over all $y \in T^m$.

The Algorithm: Given an oracle access to a circuit $C(x_1, \ldots, x_n)$ of size $s \leq n^c$ (for some constant c) and depth d' computing a polynomial $f(x_1, \ldots, x_n)$ with individual degrees at most $r = \text{polylog}(n)$ we simply evaluate C on all the elements of the hitting set. If all the evaluations are zero we output "zero", otherwise we say "non-zero".

Correctness: Denote

$$F(y) = F(y_1, \ldots, y_l) = f(p_m(y|_{S_1}), \ldots, p_m(y|_{S_n})).$$

Notice, that the algorithm actually computes the evaluation of F over all the elements of T^m. Clearly, if $f \equiv 0$ then the algorithm will output zero. Therefore it is enough to show that if $f \not\equiv 0$ then $F(y) \neq 0$ for some $y \in T^m$. As $\deg(F) \leq n \cdot m \cdot r$, the brute force algorithm (Algorithm 2.3) guarantees that if $F \not\equiv 0$ then there will be some $y \in T^m$ for which $F(f) \neq 0$. Thus, it is enough to prove that $F \not\equiv 0$.

Suppose in contradiction that $f \not\equiv 0$ and $F(y) \equiv 0$. By applying Lemma 4.1 (using the bound $|\mathbb{F}| \geq 2^m \geq n^2$) we get that p_m can be computed by a circuit of size $(s \cdot m^r)^a$ and depth $d' + 5$ for absolute constant a. Setting $A = a \cdot c + 1$ we have that $s^a \leq n^{a \cdot c} \leq 2^{\frac{a \cdot c}{A} m^{\epsilon/3}} < 2^{m^{\epsilon/3}}$ and that $m^{r \cdot a} \leq m^{\frac{a}{A} \cdot m^{\epsilon/3}} \leq 2^{m^{\epsilon/3} \cdot \log(m)} \leq 2^{m^{\epsilon/2}}$, (using the inequality $r \leq \frac{1}{A} \cdot m^{\epsilon/3}$). Therefore the total circuit size is bounded by $(s \cdot m^r)^a \leq 2^{m^\epsilon}$, which violates the lower bound assumption on p_m as $d' + 5 = d$.

Running time: Clearly the running time is equivalent to the time required for constructing the hitting set. Step 1. of the construction can be done in time $\text{poly}(n, 2^l) = n^{\text{polylog}(n)}$ by Lemma 2.1. In Step 2. for each S_i we have to evaluate $p_m(y|_{S_i})$ on $(nmr + 1)^m$ inputs. This requires, in the worst case, $2^m \cdot (nmr + 1)^m = n^{\text{polylog}(n)}$ time. □

5 Implications of derandomizing CPIT$^d(\mathbb{Q})$

We now give a sketch of the proof of Theorem 3. The proof is an exact analog of the proof of Theorem 4.1 of [KI04]. Their proof has the following structure (in brackets we give the modification needed to prove the observation). Assume that NEXP $\subseteq$ P/poly. Then by previous results (a combination of [Tod91] and [IKW02]) we get that coNEXP $\subseteq$ P$^{0-1\text{Perm}}$, where $0-1$Perm is the (boolean) language (M, v) of all 0/1 matrices M and the value of their permanent v (given in binary). We now want to show that if the other two conditions hold then P$^{0-1\text{Perm}} \subseteq$ NSUBEXP. Combining the two containments we get that coNEXP $\subseteq$ NSUBEXP, which is a contradiction as a simple diagonalization shows.

Thus, to conclude the argument we need to show that if Permanent has polynomial size (depth d) circuits and CPIT($\mathbb{Q}$) (CPIT$^d(\mathbb{Q})$) is in NSUBEXP, then P$^{0-1\text{Perm}} \subseteq$ NSUBEXP. Indeed, if the Permanent has polynomial size (depth d) circuits then we can guess such a circuit C. Now, using the simple fact that the permanent of an $k \times k$ matrix for any $k \leq n$ can be computed by a circuit for an $n \times n$ Permanent we can assume w.l.o.g. that we have a circuit C_k for every $k \leq n$ that should compute the $k \times k$ Permanent. We now write the following identities.

$$C_1(x) = x,$$

$$C_k(X^{(k)}) = \sum_{i=1}^{k} x_{1,i} \cdot C_{k-1}(X_i^{(k)}),$$

where $X^{(k)} = (x_{i,j})_{i,j \in [k]}$ is a $k \times k$ matrix and $X_i^{(k)}$ is its i-th minor along the first row (that is, the matrix obtained from deleting the first row and the i-th column). It is clear that $C = C_n$ computes the Permanent if and only if all the equalities are satisfied. As C is a polynomial size (depth d)

circuit then so are each of the circuits

$$C_k(X^{(k)}) - \sum_{i=1}^{k} x_{1,i} \cdot C_{k-1}(X_i^{(k)})$$

(as the Permanent is an irreducible polynomial, we can assume w.l.o.g. that the top gate of the circuit C is an addition gate and so the depth remains the same). Thus, by running our NSUBEXP PIT algorithm we can verify that C indeed computes the Permanent. As we just gave a NSUBEXP algorithm for computing the Permanent we get that $\mathrm{P}^{0-1\mathrm{Perm}} \subseteq$ NSUBEXP, as required.

6 Concluding remarks

We have demonstrated a hardness vs. randomness tradeoff for constant depth circuits. However, our understanding of this tradeoff is far from being complete. Our analysis shows that a hardness assumption implies a quasi-polynomial time algorithm for $\mathrm{CPIT}_r^d(\mathbb{F})$ only for r (the bound on the individual degrees) that is poly-logarithmic. The existence of a quasi-polynomial time algorithm for higher degrees (based on some hardness assumption) remains open.

Another problem of independent interest is the root finding problem: given a polynomial $P(x, y)$ computed by a small constant depth circuit, can we find constant depth circuits for the roots of P efficiently? Here we showed that if the field is large, then as long as the degree of y in P is small, the answer to this question is affirmative. For example, when the degree of y is constant, we can find constant depth circuits for the roots of P in polynomial time, with high probability. However, this algorithm is not efficient when the degree of y is large.

Acknowledgement

We would like to thank Gil Alon, Prahladh Harsha and Chris Umans for helpful discussions. We also wish to thank the anonymous reviewer for useful suggestions. We are grateful to Ketan Mulmuley for pointing out an error in an earlier proof of Lemma 3.1.

References

[AB03] M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *JACM*, 50(4):429–443, 2003.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[AM07] V. Arvind and P. Mukhopadhyay. The Monomial Ideal Membership Problem and Polynomial Identity Testing. In *Proceedings of the 18th ISAAC*, pages 800–811 2007.

[AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual FOCS*, pages 67–75, 2008.

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In *Proceedings of the 20th Annual STOC*, pages 301–309, 1988.

[BS83] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[CRS95] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM J. on Computing*, 24(5):1036–1050, 1995.

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

[GK98] D. Grigoriev and M. Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *Proceedings of the 30th Annual STOC*, pages 577–582, 1998.

[GKS90] D. Grigoriev, M. Karpinski, and M. F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. on Computing*, 19(6):1059–1063, 1990.

[GR00] D. Grigoriev and A. A. Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Applicable Algebra in Engineering, Communication and Computing*, 10(6):465–487, 2000.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. of Computer and System Sciences*, 65(4):672–694, 2002.

[ISW01] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the 32nd STOC Conference*, pages 1–10, 2001.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the xor lemma. In *Proceedings of the 29th STOC*, pages 220–229, 1997.

[Kal89] E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness in Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. 1989.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual STOC*, pages 216–223, 2001.

[KS07a] Z. Karnin and A. Shpilka. Deterministic Black Box Polynomial Identity Testing of Depth-3 Arithmetic Circuits with Bounded Top Fan-in. In *Proceedings of the 23rd Annual CCC*, pages 280–291, 2008.

[KS09] N. Kayal and S. Saraf. Blackbox Polynomial Identity Testing for Depth 3 Circuits. Manusript. 2009.

[KS07b] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

[LV98] D. Lewin and S. Vadhan. Checking polynomial identities over any field: Towards a derandomization? In *Proceedings of the 30th Annual STOC*, pages 428–437, 1998.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Pud94] P. Pudlak. Communication in bounded depth circuits. *Combinatorica*, 14(2):203–216, 1994.

[Raz07] R. Raz. Elusive functions and lower bounds for arithmetic circuits.
In *Proceedings of the 40th Annual STOC*, pages 711–720, 2008.

[RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.* 5, 3, 207-388. 2010.

[SS09] N. Saxena and C. Seshadri. An Almost Optimal Rank Bound for Depth-3 Identities. *CCC '09 (to appear)*. 2009.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *JACM*, 27(4):701–717, 1980.

[Str73] V. Strassen. Vermeidung von divisionen. *J. of Reine Angew. Math.*, 264:182–202, 1973.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. of Computer and System Sciences*, 62, 2001.

[SU05] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *JACM*, 52(2):172–216, 2005.

[SW01] A. Shpilka and A. Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001.

[Tod91] S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM J. on Computing*, 20(5):865–877, 1991.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation*, pages 216–226. 1979.