

# Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits

Zeev Dvir \*

Amir Shpilka<sup>†</sup>

Amir Yehudayoff<sup>‡</sup>

## 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  $f(x_1, \dots, x_m)$  that cannot be computed by a depth  $d$  arithmetic circuit of small size then there exists an efficient deterministic algorithm to test whether a given depth  $d - 8$  circuit is identically zero or not (assuming the individual degrees of the tested circuit are not too high). 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 [KI04] 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  $P(x_1, \dots, x_n, y) \equiv 0$  and shows that if  $P$  has a circuit of depth  $d$  and size  $s$  and if the polynomial  $f(x_1, \dots, x_n)$  satisfies  $P(x_1, \dots, x_n, f(x_1, \dots, x_n)) \equiv 0$  then  $f$  has a circuit of depth  $d + 3$  and size  $O(s \cdot r + m^r)$ , where  $m$  is the total degree of  $f$  and  $r$  is the degree of  $y$  in  $P$ .

In the other direction we observe that the methods of [KI04] imply that if we can derandomize polynomial identity testing for bounded depth circuits then NEXP does not have bounded depth arithmetic circuits. That is, either  $\text{NEXP} \not\subseteq P/\text{poly}$  or the Permanent is not computable by polynomial size bounded depth arithmetic circuits.

---

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

<sup>†</sup>Faculty of Computer Science, The Technion, Haifa, Israel. [shpilka@cs.technion.ac.il](mailto:shpilka@cs.technion.ac.il). Research supported by the Israel Science Foundation (grant number 439/06).

<sup>‡</sup>Department of Computer Science, Weizmann institute of science, Rehovot, Israel. [amir.yehudayoff@weizmann.ac.il](mailto: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. Using the well-known Schwarz-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 work 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 than 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] 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 their degrees (see the next section ‘our results’ 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 3.1) that bounds the bounded depth complexity of a polynomial root  $y = f(x_1, \dots, x_n)$  of an equation  $P(x_1, \dots, x_n, y) \equiv 0$  in term of the bounded depth complexity of  $P$ . This theorem replaces the polynomial factorization algorithm of [Kal89] in the proof of [KI04]. We note that the methods of [Kal89] does not seem to work in the bounded depth case since the circuit for the root  $f(x)$  is always of large depth, even if  $P$  has a bounded depth circuit.

## 1.1 Our results

We proceed by giving the definitions necessary to state our results formally. We start by defining three variants of the PIT problem. The problems are ordered from the most general one to the most restricted one. Our results will 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  $C(x_1, \dots, x_n)$  of size  $\text{poly}(n)$  over a field  $\mathbb{F}$ .
- *Output:* Does  $C(x) \equiv 0$  ?

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

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

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

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

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 will be 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).** Notice that the definition of CPIT<sub>r</sub><sup>d</sup>( $\mathbb{F}$ ) 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 (we leave the details to the reader). 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  $p_m \in \mathbb{F}[x_1, \dots, x_m]$  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  $p_m(x)$ .*

**Definition 1.3 (Hardness against circuits).** *Let  $\mathbb{F}$  be a field and let  $\mathbb{E}$  be an extension field of  $\mathbb{F}$ . Let  $p \in \mathbb{F}[x_1, \dots, x_m]$  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). We note that the constant  $\delta$  appearing in the two theorems can be bounded explicitly by following the steps of the proof closely. A bound of  $\delta \leq 8$  is easy to be convinced of, though a tighter bound could be achieved under certain assumptions on the given circuits.

**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  $\mathbb{Q}[x_1, \dots, x_m]$  and is  $(2^{m^\epsilon}, d, \mathbb{Q})$ -hard. Then the problem  $CPIT_{\text{polylog}(n)}^{d'}(\mathbb{Q})$  can be solved deterministically in time  $n^{\text{polylog}(n)}$ , where  $d' = d - \delta$  for some absolute constant  $\delta$ .*

**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  $\mathbb{F}[x_1, \dots, x_m]$  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  $CPIT_{\text{polylog}(n)}^{d'}(\mathbb{F})$  can be solved deterministically in time  $n^{\text{polylog}(n)}$ , where  $d' = d - \delta$  for some absolute constant  $\delta$ .*

**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  $CPIT_{\text{polylog}(n)}^{d'}(\mathbb{F})$  with  $CPIT_{n^{o(1)}}^{d'}(\mathbb{F})$  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).*

Finally, we observe that the bounded depth analog of Theorem 4.1 of [KI04] (stated below) also holds.

**Theorem (4.1 in [KI04]).** *The following three assumptions cannot be simultaneously true.*

1.  $\text{NEXP} \subseteq \text{P/poly}$ ,
2. Permanent is computable by polynomial size arithmetic circuits<sup>1</sup> over  $\mathbb{Q}$ ,

<sup>1</sup>Actually [KI04] allow the circuits to have divisions, but our circuits do not use divisions.

3.  $\text{CPIT}(\mathbb{Q})$  is in  $\text{NSUBEXP}$ .

The bounded depth analog of the theorem is given in the following observation.

**Observation 1.** *The following three assumptions cannot be simultaneously true.*

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

## 1.2 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 Observation 1 is given in Section 5.

## 2 Preliminaries

### 2.1 Notations

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

### 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  $S_1, \dots, S_n \subset [l]$  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  $\text{poly}(n, 2^l)$ .

### 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  $f \in \mathbb{F}[x_1, \dots, x_n]$  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  $C(x_1, \dots, x_n)$  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 test if  $C(a) = 0$ . 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  $f(x) \in \mathbb{F}[x_1, \dots, x_n]$  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  $H_0[f], \dots, H_m[f]$ .*

*Proof.* Let  $z$  be a new formal variable and define

$$g(x_1, \dots, x_n, z) = f(x_1 \cdot z, \dots, x_n \cdot z). \quad (1)$$

Notice that

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

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  $c_0, \dots, c_m \in \mathbb{F}$  be  $m + 1$  distinct elements. Let  $\Gamma$  be an  $(m + 1) \times (m + 1)$  matrix whose  $i$ ’th row is  $(1, c_i, c_i^2, \dots, c_i^m)$ . Let  $\alpha$  be the column vector  $(H_0[f], H_1[f], \dots, H_m[f])^t$  and  $\beta = (g(x, c_0), g(x, c_1), \dots, g(x, c_m))^t$ . Using Eq. 2 we see that  $\beta = \Gamma \cdot \alpha$  and so, since  $\Gamma$  is invertible, we have  $\alpha = \Gamma^{-1} \cdot \beta$ . Computing all the entries of  $\beta$  (in parallel) can be done in depth  $d$  and size  $O(s \cdot m)$  using the identity in Eq. 1. Now, computing the entries of  $\alpha$  can be done by adding another layer of addition gates computing the linear transformation  $\Gamma^{-1}$ . This increases the depth by one. Notice that if the top gate in the original circuit was an addition gate then the depth doesn’t increase at all.  $\square$

## 3 Roots of Equations With Polynomial Coefficients

The following theorem shows that a solution  $y = f(x_1, \dots, x_n)$  of a polynomial equation  $P(x_1, \dots, x_n, y) = 0$  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.

**Theorem 3.1.** *Let  $n, s, r, m, t, d$  be integers such that  $s \geq n$ . Let  $\mathbb{F}$  be a field which has at least  $\max\{mt + 1, r + 1\}$  elements. Let  $P(x, y) \in \mathbb{F}[x_1, \dots, x_n, y]$  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*

$f(x) \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial with  $\deg(f) = m$  such that  $P(x, f(x)) \equiv 0$ . Then  $f(x)$  can be computed by a circuit of size  $s' = \text{poly}(s, m^r)$  and depth<sup>2</sup>  $d' = d + O(1)$  over  $\mathbb{F}$ .

The heart of the proof of Theorem 3.1 is the following lemma which 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 3.1 we will use only the last approximation (when  $k = m$ ) to derive a circuit for  $f(x)$ . Notice that Lemma 3.2 does not give us any information about the degree of the approximating polynomial. We will have to take care of this fact later, in the proof of Theorem 3.1.

**Lemma 3.2.** *Let  $\mathbb{F}$  be a field, let  $P \in \mathbb{F}[x_1, \dots, x_n, y]$  be such that  $\deg_y(P) = r$  and let  $f \in \mathbb{F}[x_1, \dots, x_n]$  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  $Q_k \in \mathbb{F}[z_0, \dots, z_r]$  such that*

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

We defer the proof of Lemma 3.2 to section 3.2 and proceed to give the proof of Theorem 3.1.

### 3.1 Proof of Theorem 3.1

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

**Claim 3.3.** *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  $O(s \cdot r)$  and depth  $d + 3$ .*

*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 (namely, evaluating  $P(x, \xi)$  for  $r + 1$  values of  $\xi$  and interpolating) we get that there exists a circuit of size  $O(s \cdot r)$  and depth  $d + 1$  with  $r + 1$  outputs computing  $C_0(x), \dots, C_r(x)$ . Now, each partial derivative is a linear combination of products  $C_j(x) \cdot y^i$ , and so we can compute them with an increase of at most two in the depth of the circuit.  $\square$

**Part I - Preparations:** Observing Claim 3.3 we can assume w.l.o.g that

$$\frac{\partial P}{\partial y}(x, f(x)) \neq 0. \tag{3}$$

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  $\frac{\partial \tilde{P}}{\partial y}(x, f(x)) \neq 0$ . The loss in the size and depth incurred by Claim 3.3 is too small to be a problem. Therefore, there exists a point  $x^0 \in \mathbb{F}^n$  such that  $\frac{\partial P}{\partial y}(x^0, f(x^0)) \neq 0$  (we use the fact that  $|\mathbb{F}| > t \cdot m \geq \deg(P(x, f(x)))$ ). We can further assume w.l.o.g that  $x^0 = 0$ , for 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 Eq. 3, on the derivative not being identically zero is maintained by this transformation). To conclude, we are now in a situation where

$$\frac{\partial P}{\partial y}(0, f(0)) \neq 0. \tag{4}$$

---

<sup>2</sup>Following the proof closely one can get an exact bound of  $d + 3$  on the depth of the circuit computing  $f(x)$ . This bound can be further reduced to  $d + 2$  if the top-most gate in the circuit for  $P$  happens to be an addition gate.

**Part II - Using Lemma 3.2:** Write

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

Applying Lemma 3.2 with  $k = m$  we get that there exists a polynomial  $Q \in \mathbb{F}[z_0, \dots, z_r]$  such that

$$f(x) \equiv H_{\leq m}[f(x)] \equiv H_{\leq m}[Q(C_0(x), \dots, C_r(x))]. \quad (5)$$

Let  $z^* = (C_0(0), \dots, C_r(0))$  and suppose  $\deg(Q) = M$ . Let us define  $I_M \triangleq \{(\alpha_0, \dots, \alpha_r) \in \mathbb{N}^{r+1} \mid \sum_i \alpha_i \leq M\}$ . We can expand  $Q(z)$  around the point  $z^*$  to get an expression of the form

$$Q(z) = \sum_{\alpha \in I_M} Q_\alpha \cdot \prod_{i=0}^r (z_i - z_i^*)^{\alpha_i}.$$

Substituting  $z_i = C_i(x)$  and using Eq. 5 we get

$$f(x) = H_{\leq m} \left[ \sum_{\alpha \in I_M} Q_\alpha \cdot \prod_{i=0}^r (C_i(x) - z_i^*)^{\alpha_i} \right]. \quad (6)$$

We now observe that for every  $0 \leq i \leq r$  the polynomial  $C_i(x) - z_i^* = C_i(x) - C_i(0)$  does not have a constant term and so, if we take  $\alpha = (\alpha_0, \dots, \alpha_r)$  with  $\sum_i \alpha_i > m$  the product  $\prod_{i=0}^r (C_i(x) - z_i^*)^{\alpha_i}$  will only contain monomials of degree larger than  $m$ . Following this observation we conclude that we can ‘throw away’ all monomials of degree larger than  $m$  in the r.h.s of Eq. 6 and still have an identity. Therefore,

$$f(x) = H_{\leq m} \left[ \sum_{\alpha \in I_m} Q_\alpha \cdot \prod_{i=0}^r (C_i(x) - z_i^*)^{\alpha_i} \right]. \quad (7)$$

**Part III - Finding a circuit for  $f(x)$ :** We will now use Eq. 7 to prove the existence of a small bounded depth circuit for  $f(x)$ . Following the above notations we denote by

$$\tilde{Q}(z) = \sum_{\alpha \in I_m} Q_\alpha \cdot \prod_{i=0}^r (z_i - z_i^*)^{\alpha_i}$$

the ‘truncated’ version of  $Q$ . So  $f(x)$  is given by the homogenous parts of degree at most  $m$  in the polynomial  $g(x) \triangleq \tilde{Q}(C_0(x), \dots, C_r(x))$ . We can compute  $\tilde{Q}(z)$  using a depth 2 circuit of size  $\leq m^r$ . Observing the proof of Claim 3.3 we also see that we can compute the polynomials  $C_0(x), \dots, C_r(x)$  with a circuit of depth  $d + O(1)$  and size  $O(s \cdot r)$ . Therefore, the polynomial  $g(x)$  can be computed by a circuit of depth  $d + O(1)$  and size  $\text{poly}(s, m^r)$ . Using Lemma 2.4 we get that  $f(x)$  can be computed by a circuit of size  $\text{poly}(s, m^r)$  and depth  $d + O(1)$  (we use the fact that  $|\mathbb{F}| > t \cdot m \geq \deg(g)$ ).

### 3.2 Proof of Lemma 3.2

We will prove the lemma by induction on  $k$ . For  $k = 0$  the lemma is trivial. Suppose that we have proven the lemma for  $k - 1$ . That is, there exists a polynomial  $Q_{k-1}(z)$  such that  $H_{\leq k-1}[f(x)] = H_{\leq k-1}[Q_{k-1}(C_0(x), \dots, C_r(x))]$ . To simplify notations let us denote  $g(x) = Q_{k-1}(C_0(x), \dots, C_r(x))$ .

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  $H_{\leq k-1}[f(x)] = H_{\leq k-1}[g(x)]$  and that  $P(0, g(0)) = P(0, f(0)) = \xi_0 \neq 0$  (the constant term is the same in  $f$  and in  $g$ ). Let us denote also  $f_k = H_k[f]$  and similarly for  $g$ .

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

Notice now that the polynomial  $\{f_k(x) - g_k(x)\} \cdot P'(x, g(x))$  contains only monomials of degree  $k$  and above (if  $f_k(x) \equiv g_k(x)$  then there is nothing to prove) and that in order to get the homogenous part of degree  $k$  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

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

Rearranging we get

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

Notice that the polynomial  $g(x) - (1/\xi_0) \cdot P(x, g(x))$  can be written as a polynomial in the  $C_i$ 's and so we are done.  $\square$

## 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  $n, r, s$  be integers and let  $f \in \mathbb{F}[x_1, \dots, x_n]$  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  $S_1, \dots, S_n \subset [l]$  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  $p \in \mathbb{F}[z_1, \dots, z_m]$  be a multilinear polynomial such that*

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

*Then  $p(z)$  can be computed by a circuit of size  $\leq (s \cdot m^r)^a$  and depth  $d + b$  over  $\mathbb{F}$ , where  $a$  and  $b$  are absolute constants. 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:

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

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

$$F_i(x, y) \not\equiv 0 \quad \text{and} \quad F_{i+1}(x, y) \equiv 0. \quad (8)$$

We would like to fix all the variables  $x_{i+2}, \dots, x_n$  (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 Eq. 8 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

$$\begin{aligned} \tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \dots, q_i(y|_{S_i \cap S_{i+1}}), x_{i+1}) &\not\equiv 0 \\ \tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \dots, q_i(y|_{S_i \cap S_{i+1}}), p(y|_{S_{i+1}})) &\equiv 0, \end{aligned} \quad (9)$$

where  $\tilde{f}$  is simply  $f$  with the variables  $x_{i+2}, \dots, x_n$  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 Eq. 9 as follows:

$$\begin{aligned} g(z_1, \dots, z_m, w) &\not\equiv 0 \\ g(z_1, \dots, z_m, p(z_1, \dots, z_m)) &\equiv 0, \end{aligned} \quad (10)$$

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

$$\tilde{f}(q_1(y|_{S_1 \cap S_{i+1}}), \dots, q_i(y|_{S_i \cap S_{i+1}}), x_{i+1}) \quad (11)$$

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

Our final step is to apply Theorem 3.1 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  $\leq s^{a'}$  and depth  $d + b'$  for absolute constants  $a'$  and  $b'$ .*

*Proof.* Since  $g(z, w)$  has the same circuit complexity as the expression in Eq. 11 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).  $\square$

Notice that  $\deg_w(g) = \deg_{x_{i+1}}(f) \leq r$ . We can therefore use Theorem 3.1 to get that  $p(z)$  has a circuit of depth  $d + b$  and size  $\leq (s \cdot m^r)^a$ . If  $\mathbb{F}$  is finite then the conditions of Theorem 3.1 are satisfied since  $|\mathbb{F}| \geq n^2 > \max\{r + 1, \deg(g) \cdot \deg(p) + 1\}$ .  $\square$

## 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  $p \in \mathbb{F}[z_1, \dots, z_m]$  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 Algorithm:** We are given a circuit  $C(x_1, \dots, x_n)$  of size  $s \leq n^c$  (for some constant  $c$ ) and depth  $d'$  computing a polynomial  $f(x_1, \dots, x_n)$  with individual degrees at most  $r = \text{polylog}(n)$ . We fix  $A$  to be some large number 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  $\epsilon$  is such that  $p_m$  is  $(2^{m^\epsilon}, d, \mathbb{F})$ -hard). The algorithm proceeds in the following four steps:

1. Construct a design  $S_1, \dots, S_n \subset [l]$  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. Construct a circuit of size  $2^m$  computing the polynomial  $p_m(z_1, \dots, z_m)$  (this task is trivial given the list of coefficients of  $p_m$ ).
3. Construct a circuit for the polynomial

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

by composing  $C$  with  $n$  copies of the circuit obtained for  $p_m$  in the previous step.

4. Check if  $F(y) \equiv 0$  using the brute force (BF) algorithm (Algorithm 2.3) providing a bound of  $n \cdot m \cdot r$  on the degree of  $F(y)$ . Return the answer of the BF algorithm.

**Correctness:** Clearly, if  $f \equiv 0$  then the polynomial  $F$  constructed in step 3 is also zero, and so the BF algorithm will return the correct answer, provided that the degree of  $F(y)$  is indeed at most  $n \cdot m \cdot r$ . This is true since the degree of  $f$  is at most  $n \cdot r$  and the degree of  $p_m$  is at most  $m$  (it is multilinear). Therefore it is enough to show that if  $f \not\equiv 0$  then  $F(y) \not\equiv 0$ .

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  $d' + b$  for absolute constants  $a$  and  $b$ . 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  $p_m$ , provided that  $d' + b \leq d$  which will hold if we take the constant  $\delta$  in the theorem to be equal to  $b$ .

**Running time:** Step 1. of the algorithm can be done in time  $\text{poly}(n, 2^l) = n^{\text{polylog}(n)}$  by Lemma 2.1. Step 2. can be done in time  $2^{O(m)} = n^{\text{polylog}(n)}$  since  $p_m$  is weakly explicit. Step 3. can be done in time roughly  $n \cdot 2^m = n^{\text{polylog}(n)}$ . Step 4. takes  $(nmr + 1)^m = n^{\text{polylog}(n)}$  evaluations of the circuit for  $F$ , each taking  $n^{\text{polylog}(n)}$  time.  $\square$

## 5 Implications of Derandomizing $\text{CPIT}^d(\mathbb{Q})$

We now give a sketch of the proof of Observation 1. 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  $\text{NEXP} \subseteq \text{P/poly}$ . Then by previous results (a combination of [Tod91] and [IKW02]) we get that  $\text{coNEXP} \subseteq \text{P}^{0-1\text{Perm}}$ , where  $0-1\text{Perm}$  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  $\text{P}^{0-1\text{Perm}} \subseteq \text{NSUBNEXP}$ . Combining the two containments we get that  $\text{coNEXP} \subseteq \text{NSUBNEXP}$ , 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  $\text{CPIT}(\mathbb{Q})$  ( $\text{CPIT}^d(\mathbb{Q})$ ) is in  $\text{NSUBEXP}$ , then  $\text{P}^{0-1\text{Perm}} \subseteq \text{NSUBNEXP}$ . 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  $\text{NSUBEXP}$  PIT algorithm we can verify that  $C$  indeed computes the Permanent. As we just gave a  $\text{NSUBEXP}$  algorithm for computing the Permanent we get that  $\text{P}^{0-1\text{Perm}} \subseteq \text{NSUBNEXP}$ , as required.

## Acknowledgement

We would like to thank Prahladh Harsha, Chris Umans and Gil Alon for helpful discussions.

## 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 ideal membership problem and polynomial identity testing. *ECCC Report TR07-095*, 2007.

- [BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial 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. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. Manuscript, 2007.
- [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*. Akademie-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.
- [MUV87] 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. *Manuscript*, 2007.
- [RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005.
- [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.