Tensor-Rank and Lower Bounds for Arithmetic Formulas

Ran Raz*
Weizmann Institute

Abstract

We show that any explicit example for a tensor $A : [n]^r \to \mathbb{F}$ with tensor-rank $\geq n^{r\cdot(1-o(1))}$, where $r = r(n) \leq \log n / \log\log n$ is super-constant, implies an explicit super-polynomial lower bound for the size of general arithmetic formulas over $\mathbb{F}$. This shows that strong enough lower bounds for the size of arithmetic formulas of depth 3 imply super-polynomial lower bounds for the size of general arithmetic formulas.

One component of our proof is a new approach for homogenization and multilinearization of arithmetic formulas, that gives the following results:

We show that for any n-variate homogeneous polynomial f of degree r, if there exists a (fanin-2) formula of size s and depth d for f then there exists a homogeneous formula of size $O\left(\binom{d+r+1}{r} \cdot s\right)$ for f. In particular, for any $r \leq O(\log n)$, if there exists a polynomial size formula for f then there exists a polynomial size homogeneous formula for f. This refutes a conjecture of Nisan and Wigderson [NW95] and shows that super-polynomial lower bounds for homogeneous formulas for polynomials of small degree imply super-polynomial lower bounds for general formulas.

We show that for any n-variate set-multilinear polynomial f of degree r, if there exists a (fanin-2) formula of size s and depth d for f then there exists a set-multilinear formula of size $O\left((d+2)^r \cdot s\right)$ for f. In particular, for any $r \leq O(\log n / \log\log n)$, if there exists a polynomial size formula for f then there exists a polynomial size set-multilinear formula for f. This shows that super-polynomial lower bounds for set-multilinear formulas for polynomials of small degree imply super-polynomial lower bounds for general formulas.

*ran.raz@weizmann.ac.il, Research supported by the Israel Science Foundation (ISF), the Binational Science Foundation (BSF) and the Minerva Foundation.

1 Introduction

1.1 Arithmetic Formulas

Let $\mathbb{F}$ be a field and let $\{x_1, \ldots, x_n\}$ be a set of input variables. An *arithmetic formula* is a directed tree whose edges are directed from the leaves to the root. Every leaf of the tree is labeled with either an input variable or a field element. Every other node of the tree is labeled with either $+$ or $\times$; in the first case the node is a *sum gate* and in the second case a *product gate*. The *size* of a formula is the number of edges in it. The *depth* of a formula is the length of the longest directed path in it[1]. The *fanin* of a gate is its in-degree.

Every node of an arithmetic formula computes a polynomial in the ring $\mathbb{F}[x_1, \ldots, x_n]$ as follows. A leaf just computes the input variable or field element that labels it. A sum gate computes the sum of the polynomials computed by its children. A product gate computes the product of the polynomials computed by its children. The *output* of the formula is the polynomial computed by the root. The root of the formula is also called the *output node*.

An *arithmetic circuit* is defined in the same way as an arithmetic formula, except that the underlying graph is a general directed acyclic graph (rather than a directed tree). For simplicity, we assume that every circuit has exactly one output node.

Proving super-polynomial lower bounds for the size of arithmetic circuits and formulas (for explicit polynomials) is one of the most interesting and most challenging open problems in computational complexity. Such lower bounds are only known for restricted cases. For example, super-polynomial lower bounds were proved for non-commutative formulas [Nis91], for depth 3 formulas over finite fields [GK98, GR98], and for multilinear formulas [R04a, R04b].

For an excellent recent survey on arithmetic circuits and formulas, see [SY10].

1.2 Homogeneous Formulas

A polynomial f in the ring $\mathbb{F}[x_1, \ldots, x_n]$ is *homogeneous* if all the monomials that occur in f are of the same degree. An arithmetic formula (or circuit) is *homogeneous* if the polynomial computed by each of its nodes is homogeneous.

A standard (and straightforward) homogenization technique by Strassen [Str73] shows that for any homogeneous polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree r, if there exists a formula of size s for f then there exists a homogeneous formula of size $\text{poly}(s^{\log r})$ for f. It was conjectured (see for example [NW95]) that this technique is optimal for every degree r. We show that this is not the case. In particular, we show that for any homogeneous polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree $r \leq O(\log n)$, if there exists a polynomial size formula for f then there exists a polynomial size homogeneous formula for f. Thus, super-polynomial lower bounds for homogeneous formulas for polynomials of degree up to $O(\log n)$ imply super-

[1] When counting the depth of a formula, it is customary not to count *scalar-products*, that is, product gates of fanin 2 that at least one of their children is a leaf labeled by a field element.

polynomial lower bounds for general arithmetic formulas. Our approach can be viewed as a tighter analysis of the standard technique.

1.3 Set-Multilinear Formulas

Let $X_1, \ldots, X_r$ be disjoint sets of variables. A polynomial f over the set of variables $X_1 \cup \ldots \cup X_r$ is *set-multilinear* in the sets $X_1, \ldots, X_r$ if every monomial that occurs in f is multilinear and contains exactly one variable from each set X_i. An arithmetic formula (or circuit) is *set-multilinear* in the sets $X_1, \ldots, X_r$ if the polynomial computed by its output node is set-multilinear in the sets $X_1, \ldots, X_r$ and the polynomial computed by each of its other nodes is set-multilinear in a subset of $\{X_1, \ldots, X_r\}$.

A standard (and straightforward) multilinearization technique shows that for any set-multilinear polynomial f over the sets $X_1, \ldots, X_r$, if there exists a formula of size s for f then there exists a set-multilinear formula of size $\text{poly}(s^r)$ for f. We show that this technique is not optimal. In particular, we show that for any set-multilinear polynomial f of degree $r \leq O(\log n / \log\log n)$, if there exists a polynomial size formula for f then there exists a polynomial size set-multilinear formula for f. Thus, super-polynomial lower bounds for set-multilinear formulas for polynomials of degree up to $O(\log n / \log\log n)$ imply super-polynomial lower bounds for general arithmetic formulas. Our approach can be viewed as a tighter analysis of the standard technique.

Set-multilinear formulas were first studied in [NW95]. Super-polynomial lower bounds for multilinear formulas (that are more general than set-multilinear formulas) were proved in [R04a, R04b] (see also [Aar04, RSY07, RY08a, RY08b]). These techniques however do not give super-polynomial lower bounds for polynomials of very small degree.

1.4 Tensor-Rank

A tensor $A : [n]^r \to \mathbb{F}$ is of rank 1 if there exist r vectors $a_1, \ldots, a_r : [n] \to \mathbb{F}$ such that $A = a_1 \otimes \ldots \otimes a_r$ (where $\otimes$ denotes tensor product, that is, A is defined by $A(i_1, \ldots, i_r) = a_1(i_1) \cdots a_r(i_r)$). More generally, the *tensor-rank* of A is the minimal k such that there exist k tensors $A_1, \ldots, A_k : [n]^r \to \mathbb{F}$ of rank 1 such that $A = \sum_{i=1}^{k} A_i$. This is a natural generalization of matrix-rank.

Given a tensor $A : [n]^r \to \mathbb{F}$ and r sets of variables $X_1, \ldots, X_r$, where $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, one can define the set-multilinear polynomial f_A as follows

$$f_A(x_{1,1}, \ldots, x_{r,n}) = \sum_{i_1,\ldots,i_r \in [n]} A(i_1, \ldots, i_r) \cdot \prod_{j=1}^{r} x_{j,i_j}$$

A beautiful well known and straightforward insight, going back to [Str73] (see [Gat88] for a survey), shows that the tensor-rank of tensors of order $r = 3$ is very related to *bilinear complexity* and hence also to general arithmetic circuit complexity. In particular (using

also [BS83]), if the tensor-rank of $A : [n]^3 \to \mathbb{F}$ is k then the smallest circuit for f_A is of size $\Omega(k)$. Thus, one can prove lower bounds for arithmetic circuit size by proving lower bounds for tensor-rank. Note, however, that the tensor-rank of $A : [n]^3 \to \mathbb{F}$ is bounded by $O(n^2)$. Hence, this approach can only give lower bounds of up to $\Omega(n^2)$ for arithmetic circuit size. We note that the tensor-rank of most tensors $A : [n]^3 \to \mathbb{F}$ is $\Theta(n^2)$. However, to date, no lower bound better than $\Omega(n)$ is known for the tensor-rank of any explicit tensor $A : [n]^3 \to \mathbb{F}$. Lower bounds of $3n - O(\log n)$ were recently proved in [AFT11].

Here we consider tensors $A : [n]^r \to \mathbb{F}$ where $r = r(n)$ is super-constant and satisfies $r \leq O(\log n / \log \log n)$. We show that for any such A, if there exists an arithmetic formula of size n^c for f_A then the tensor-rank of A is at most $n^{r \cdot (1-2^{-O(c)})}$. Thus, a lower bound of $n^{r \cdot (1-o(1))}$ for the tensor-rank of A implies a super-polynomial lower bound for the size of any arithmetic formula for[2] f_A.

Since the tensor-rank of A corresponds to computations of f_A by depth-3 (set-multilinear) formulas, our result shows that strong enough lower bounds for the size of arithmetic formulas of depth 3 imply super-polynomial lower bounds for the size of general arithmetic formulas. Previously, it was well known that strong enough lower bounds for the size of arithmetic circuits of depth 4 imply exponential lower bounds for the size of general arithmetic circuits (see for example [RY08a, R08]). Moreover, a striking recent result of Agrawal and Vinay (based on [VSBR83]) shows that *any* exponential lower bound for the size of arithmetic circuits of depth 4 implies an exponential lower bound for the size of general arithmetic circuits [AV08].

We note that it is very easy to give lower bounds of $n^{\lfloor r/2 \rfloor}$ for the tensor-rank of tensors $A : [n]^r \to \mathbb{F}$ (by taking a full-rank matrix of size $n^{\lfloor r/2 \rfloor} \times n^{\lfloor r/2 \rfloor}$). Lower bounds of $2n^{\lfloor r/2 \rfloor} + n - O(r \log n)$ for the tensor-rank of explicit tensors $A : [n]^r \to \mathbb{F}$ (for odd r) were recently proved in [AFT11]. We note also that it was proved by Håstad that computing the tensor-rank is an NP-complete problem [H89].

1.5 Preliminaries

We say that an arithmetic formula (or circuit) is of *fanin* 2 if the fanin of every gate in it is 2. We say that an arithmetic formula (or circuit) is of *product-fanin*-2 if the fanin of every product gate in it is 2. The *product-depth* of a product-fanin-2 formula (or circuit) is the maximal number of product gates along a directed path in it.

For a formula (or circuit) Φ and a node u in it, we denote by Φ_u the sub-formula of Φ rooted at u and by $\hat{\Phi}_u$ the polynomial computed by the formula Φ_u.

It is well known that for any fanin-2 formula Φ of size s, one can assume without loss of generality that the depth of Φ is $O(\log s)$. That is, there exists a formula of size $\text{poly}(s)$ and depth $O(\log s)$ that computes the same polynomial computed by Φ.

[2]Moreover, it was noted to us by Amir Yehudayoff that by the completeness of the permanent [Val79], if the tensor A is "explicit" the super-polynomial lower bound holds for the permanent.

1.6 Discussion

A well known approach, first suggested by Strassen [Str73], is to consider the tensor-product of tensors of high tensor-rank as a candidate for a larger tensor with high tensor-rank.

Let n, m, r be such that $m^r = n$, and for simplicity assume that $\mathbb{F}$ is a finite field. Let $A_1, \ldots, A_r : [m]^r \to \mathbb{F}$ be random tensors, and let $A = A_1 \otimes \ldots \otimes A_r : [n]^r \to \mathbb{F}$ be their tensor-product, defined by

$$A((i_{1,1}, \ldots, i_{1,r}), \ldots, (i_{r,1}, \ldots, i_{r,r})) = A_1(i_{1,1}, \ldots, i_{r,1}) \cdots A_r(i_{1,r}, \ldots, i_{r,r}).$$

Since $A_1, \ldots, A_r$ are random, with high probability their tensor-rank is high. If one can prove that with probability larger than 0 the tensor-rank of A is at least $n^{r \cdot (1-o(1))}$, one obtains super-polynomial lower bounds for arithmetic formulas for the polynomial f_A, where the entries of the tensors $A_1, \ldots, A_r$ are viewed as additional input variables (note that there are only $r \cdot n$ such entries).

2 Homogenization

For a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, denote by f_i the homogeneous part of f of degree i. That is, $f = \sum_i f_i$ where each f_i is a homogeneous polynomial of degree i. In the same way, for a formula (or circuit) Φ and a node u in it denote by $\hat{\Phi}_{u,i}$ the homogeneous part of degree i of the polynomial $\hat{\Phi}_u$.

Obviously, if u is a fanin-2 sum gate with children v, w then for every i

$$\hat{\Phi}_{u,i} = \hat{\Phi}_{v,i} + \hat{\Phi}_{w,i} \tag{1}$$

and if u is a fanin-2 product gate with children v, w then for every i

$$\hat{\Phi}_{u,i} = \sum_{j=0}^{i} \hat{\Phi}_{v,j} \cdot \hat{\Phi}_{w,i-j} \tag{2}$$

Let Φ be a fanin-2 circuit of size s and depth d that computes a homogeneous polynomial of degree r. There is a standard and straightforward technique to turn Φ into a fanin-2 homogeneous circuit of size $s \cdot \text{poly}(r)$ and depth $O(d \cdot \log r)$ that computes the same polynomial [Str73]. The main idea of the homogenization technique is to split every node u in the circuit into $r + 1$ nodes $u_0, \ldots, u_r$, where each node u_i in the new circuit computes the homogeneous part of degree i of the polynomial computed by u; that is, u_i computes the polynomial $\hat{\Phi}_{u,i}$. The computation of each $\hat{\Phi}_{u,i}$ is done recursively using Equation 1 and Equation 2. If u is the output node of the original circuit, the output node of the new circuit is the node u_r that computes the polynomial $\hat{\Phi}_{u,r} = \hat{\Phi}_u$ (since we assumed that the original circuit computes a homogeneous polynomial of degree r). The size of the new circuit is $s \cdot \text{poly}(r)$. When we turn the new circuit into a fanin-2 circuit its depth may increase to $O(d \cdot \log r)$ since the sum in Equation 2 is over up to $r + 1$ elements.

Let Φ be a formula of size s that computes a homogeneous polynomial of degree r. The standard homogenization technique turns Φ into a homogeneous formula of size $\text{poly}(s^{\log r})$ that computes the same polynomial, as follows. First, assume for simplicity and without loss of generality that Φ is a fanin-2 formula, and assume without loss of generality that the depth of Φ is $O(\log s)$ (see Section 1.5) - this step may increase the size of Φ polynomially. Next, use the homogenization technique described above to turn Φ into a fanin-2 homogeneous circuit of depth $O(\log s \cdot \log r)$ for the same polynomial. Finally, turn that circuit into a formula of size $2^{O(\log s \cdot \log r)} = \text{poly}(s^{\log r})$ (by duplicating every node as many times as needed, until the out-degree of every node is at most 1).

It was conjectured in [NW95] that the increase in the formula-size by a power of $O(\log r)$ in the exponent is necessary. We note also that for the special case of multilinear formulas, it was recently proved in [HY09] that for $r = \text{poly}(n)$ this is indeed the case.

2.1 A New Homogenization Theorem

Our approach can be viewed as a tighter analysis of the standard technique described above. For simplicity and without loss of generality we state and prove our theorem for fanin-2 formulas.

Theorem 1. *Let Φ be a fanin-2 formula of size s and product-depth d that computes a homogeneous polynomial of degree r. Then there exists a fanin-2 homogeneous formula Ψ of size $O\left(\binom{d+r+1}{r} \cdot s\right)$ and product-depth d that computes the same polynomial*[3]*.*

Proof. For every node u in the formula Φ, denote by $\text{path}(u)$ the set of all nodes on the directed path from u to the root (including the node u). For every node u denote by N_u the set of all functions $D : \text{path}(u) \to \{0, \ldots, r\}$ such that:

1. For every $v, w \in \text{path}(u)$ such that v is a sum gate and w is a child of v, $D(v) = D(w)$.
2. For every $v, w \in \text{path}(u)$ such that v is a product gate and w is a child of v, $D(v) \geq D(w)$.

Intuitively, a function $D \in N_u$ describes a possible progression of the degree of a monomial along the path from u to the root.

Construction of Ψ:

The formula Ψ is constructed as follows. For every node u in Φ and $D \in N_u$ we will have a node (u, D) in Ψ. Every node (u, D) will compute in Ψ the polynomial

$$\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$$

[3] In the same way, if the polynomial computed by Φ is not homogeneous and is of an arbitrary degree, and r is an integer, there exists Ψ as above that computes the homogeneous part of degree r of the polynomial computed by Φ.

that is, the homogeneous part of degree $D(u)$ of the polynomial $\hat{\Phi}_u$ computed by u in Φ. To compute $\hat{\Phi}_{u,D(u)}$, the node (u, D) will only use the outputs of nodes (v, D') such that v is a child of u and D' extends D (that is, D' agrees with D on every node on the path from u to the root). This is done inductively in a straightforward manner (using Equation 1 and Equation 2) as follows:

1. u **is a leaf:** If u is a leaf in Φ then for every $D \in N_u$ the node (u, D) will be a leaf. If u is labeled by a field element, (u, D) is labeled by the same field element if $D(u) = 0$ and by 0 if $D(u) \neq 0$. If u is labeled by an input variable, (u, D) is labeled by the same input variable if $D(u) = 1$ and by 0 if $D(u) \neq 1$. By the definitions, for every $D \in N_u$,

$$\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$$

2. u **is a sum gate:** Assume that u is a sum gate in Φ with children v, w. For every $D \in N_u$ denote by $D_v \in N_v$ the function that agrees with D on path(u) and satisfies $D_v(v) = D(u)$, and in the same way denote by $D_w \in N_w$ the function that agrees with D on path(u) and satisfies $D_w(w) = D(u)$. The node (u, D) will sum the outputs of the nodes (v, D_v) and (w, D_w). By the induction hypothesis and Equation 1 we have

$$\hat{\Psi}_{(u,D)} = \hat{\Psi}_{(v,D_v)} + \hat{\Psi}_{(w,D_w)} = \hat{\Phi}_{v,D_v(v)} + \hat{\Phi}_{w,D_w(w)} = \hat{\Phi}_{v,D(u)} + \hat{\Phi}_{w,D(u)} = \hat{\Phi}_{u,D(u)}$$

3. u **is a product gate:** Assume that u is a product gate in Φ with children v, w. For every $D \in N_u$ and $i \in \{0, \ldots, D(u)\}$ denote by $D_{v,i} \in N_v$ the function that agrees with D on path(u) and satisfies $D_{v,i}(v) = i$, and in the same way denote by $D_{w,i} \in N_w$ the function that agrees with D on path(u) and satisfies $D_{w,i}(w) = i$. The node (u, D) will compute $\hat{\Psi}_{(u,D)}$ from the outputs of the nodes $\{(v, D_{v,i}) : i \in \{0, \ldots, D(u)\}\}$ and $\{(w, D_{w,i}) : i \in \{0, \ldots, D(u)\}\}$ by the formula

$$\hat{\Psi}_{(u,D)} = \sum_{i=0}^{D(u)} \hat{\Psi}_{(v,D_{v,i})} \cdot \hat{\Psi}_{(w,D_{w,D(u)-i})}$$

   By the induction hypothesis and Equation 2 we have

$$\hat{\Psi}_{(u,D)} = \sum_{i=0}^{D(u)} \hat{\Psi}_{(v,D_{v,i})} \cdot \hat{\Psi}_{(w,D_{w,D(u)-i})} =$$

$$\sum_{i=0}^{D(u)} \hat{\Phi}_{v,D_{v,i}(v)} \cdot \hat{\Phi}_{w,D_{w,D(u)-i}(w)} = \sum_{i=0}^{D(u)} \hat{\Phi}_{v,i} \cdot \hat{\Phi}_{w,D(u)-i} = \hat{\Phi}_{u,D(u)}$$

We fix the output node of Ψ to be the node (u, D) such that u is the output node of Φ and $D \in N_u$ is the function that satisfies $D(u) = r$.

Comment: Note that in the construction above there may be nodes (u, D) that are not connected by a path to the output node of Ψ. These nodes do not contribute to the functionality of Ψ and should be removed so that the final Ψ is a tree rather than a union of trees.

Functionality of Ψ:

We proved by induction that every node (u, D) in Ψ computes the polynomial $\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$. In particular, the output node (u, D) computes the polynomial $\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)} = \hat{\Phi}_{u,r} = \hat{\Phi}_u$, which is the polynomial computed by Φ.

Properties of Ψ:

To see that Ψ is a formula note that the output of a node (v, D') is only used by a node (u, D) such that u is the parent of v and D agrees with D' on path(u), and there is at most one such node (u, D). Thus, the out-degree of every node is at most 1.

Ψ is homogeneous since (by induction) each of its nodes computes a homogeneous polynomial.

The product-depth of Ψ is the same as the product-depth of Φ since the "product-depth" of Equation 1 is 0 and the "product-depth" of Equation 2 is 1.

Finally, since the size of every N_u is bounded by $\binom{d+r+1}{r}$, the size of Ψ is at most $O\left(\binom{d+r+1}{r} \cdot s\right)$.

(Note also that we can replace every gate of fanin larger than 2 by a tree of gates of fanin 2, so that the final fanin of every gate in the formula is at most 2). □

Corollary 2. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a homogeneous polynomial of degree $\leq O(\log n)$. If there exists a polynomial size formula for f then there exists a polynomial size homogeneous formula for f.*

Proof. Let Φ be a polynomial size formula for f. Without loss of generality Φ is a fanin-2 formula of depth $O(\log n)$. The proof hence follows from Theorem 1. □

3 Multilinearization

For vectors $a, b \in \{0,1\}^r$, we say that $b \leq a$ if for every $i \in \{1, \ldots, r\}$, $b(i) \leq a(i)$.

Let $X_1, \ldots, X_r$ be disjoint sets of variables. For a vector $a \in \{0,1\}^r$ and a monomial q in the set of variables $X_1 \cup \ldots \cup X_r$, we say that q is set-multilinear of type a if it is multilinear and contains exactly one variable from each set X_i such that $a(i) = 1$ and no variables from sets X_i such that $a(i) = 0$. For a polynomial f over the set of variables $X_1 \cup \ldots \cup X_r$, and a vector $a \in \{0,1\}^r$, denote by f_a the set-multilinear part of f of type a. That is, f_a is the sum of all the set-multilinear monomials of type a that occur in f with the same coefficient as their occurrence in f.

In the same way, for a formula (or circuit) Φ and a node u in it denote by $\hat{\Phi}_{u,a}$ the set-multilinear part of type a of the polynomial $\hat{\Phi}_u$.

Obviously, if u is a fanin-2 sum gate with children v, w then for every a

$$\hat{\Phi}_{u,a} = \hat{\Phi}_{v,a} + \hat{\Phi}_{w,a} \tag{3}$$

and if u is a fanin-2 product gate with children v, w then for every a

$$\hat{\Phi}_{u,a} = \sum_{b \in \{0,1\}^r \text{ s.t. } b \leq a} \hat{\Phi}_{v,b} \cdot \hat{\Phi}_{w,a-b} \tag{4}$$

Let Φ be a fanin-2 circuit of size s and depth d that computes a set-multilinear polynomial in the sets $X_1, \ldots, X_r$. There is a standard and straightforward technique to turn Φ into a fanin-2 set-multilinear circuit of size $s \cdot \text{poly}(2^r)$ and depth $O(d \cdot r)$ that computes the same polynomial. The main idea of the multilinearization technique is to split every node u in the circuit into 2^r nodes $\{u_a\}_{a \in \{0,1\}^r}$, where each node u_a in the new circuit computes the set-multilinear part of type a of the polynomial computed by u; that is, u_a computes the polynomial $\hat{\Phi}_{u,a}$. The computation of each $\hat{\Phi}_{u,a}$ is done recursively using Equation 3 and Equation 4. If u is the output node of the original circuit, the output node of the new circuit is the node $u_{\vec{1}}$ (where $\vec{1} \in \{0,1\}^r$ denotes the vector $(1, \ldots, 1)$) that computes the polynomial $\hat{\Phi}_{u,\vec{1}} = \hat{\Phi}_u$ (since we assumed that the original circuit computes a set-multilinear polynomial in the sets $X_1, \ldots, X_r$). The size of the new circuit is $s \cdot \text{poly}(2^r)$. When we turn the new circuit into a fanin-2 circuit its depth may increase to $O(d \cdot r)$ since the sum in Equation 4 is on up to 2^r elements.

Let Φ be a formula of size s that computes a set-multilinear polynomial in the sets $X_1, \ldots, X_r$. The standard multilinearization technique turns Φ into a set-multilinear formula of size $\text{poly}(s^r)$ that computes the same polynomial, as follows. First, assume for simplicity and without loss of generality that Φ is a fanin-2 formula, and assume without loss of generality that the depth of Φ is $O(\log s)$ (see Section 1.5) - this step may increase the size of Φ polynomially. Next, use the multilinearization technique described above to turn Φ into a fanin-2 set-multilinear circuit of depth $O((\log s) \cdot r)$ for the same polynomial. Finally, turn that circuit into a formula of size $2^{O((\log s) \cdot r)} = \text{poly}(s^r)$.

3.1 A New Multilinearization Theorem

Our approach can be viewed as a tighter analysis of the standard technique described above. For simplicity and without loss of generality we state and prove our theorem for fanin-2 formulas. The technique that we use is very similar to the technique that we used in Section 2.1.

Theorem 3. *Let Φ be a fanin-2 formula of size s and product-depth d that computes a set-multilinear polynomial over the disjoint sets $X_1, \ldots, X_r$. Then there exists a fanin-2 set-multilinear formula Ψ of size $O\left((d+2)^r \cdot s\right)$ and product-depth d that computes the same polynomial.*

Proof. For every node u in the formula Φ, denote by $\text{path}(u)$ the set of all nodes on the directed path from u to the root (including the node u). For every node u denote by N_u the set of all functions $D : \text{path}(u) \to \{0,1\}^r$ such that:

1. For every $v, w \in \text{path}(u)$ such that v is a sum gate and w is a child of v, $D(v) = D(w)$.
2. For every $v, w \in \text{path}(u)$ such that v is a product gate and w is a child of v, $D(v) \geq D(w)$.

Intuitively, a function $D \in N_u$ describes a possible progression of the type of a set-multilinear monomial along the path from u to the root.

Construction of Ψ:

The formula Ψ is constructed as follows. For every node u in Φ and $D \in N_u$ we will have a node (u, D) in Ψ. Every node (u, D) will compute in Ψ the polynomial

$$\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$$

that is, the set-multilinear part of type $D(u)$ of the polynomial $\hat{\Phi}_u$ computed by u in Φ. To compute $\hat{\Phi}_{u,D(u)}$, the node (u, D) will only use the outputs of nodes (v, D') such that v is a child of u and D' extends D (that is, D' agrees with D on every node on the path from u to the root). This is done inductively in a straightforward manner (using Equation 3 and Equation 4) as follows:

1. u **is a leaf:** If u is a leaf in Φ then for every $D \in N_u$ the node (u, D) will be a leaf. If u is labeled by a field element, (u, D) is labeled by the same field element if $D(u) = \vec{0}$ and by 0 if $D(u) \neq \vec{0}$ (where $\vec{0}$ denotes the 0 vector in $\{0,1\}^r$). If u is labeled by an input variable from the set X_i, (u, D) is labeled by the same input variable if $D(u) = \vec{e_i}$ and by 0 if $D(u) \neq \vec{e_i}$ (where $\vec{e_i} \in \{0,1\}^r$ is defined by $\vec{e_i}(j) = 1$ iff $j = i$). By the definitions, for every $D \in N_u$,
$$\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$$
2. u **is a sum gate:** Assume that u is a sum gate in Φ with children v, w. For every $D \in N_u$ denote by $D_v \in N_v$ the function that agrees with D on $\text{path}(u)$ and satisfies $D_v(v) = D(u)$, and in the same way denote by $D_w \in N_w$ the function that agrees with D on $\text{path}(u)$ and satisfies $D_w(w) = D(u)$. The node (u, D) will sum the outputs of the nodes (v, D_v) and (w, D_w). By the induction hypothesis and Equation 3 we have
$$\hat{\Psi}_{(u,D)} = \hat{\Psi}_{(v,D_v)} + \hat{\Psi}_{(w,D_w)} = \hat{\Phi}_{v,D_v(v)} + \hat{\Phi}_{w,D_w(w)} = \hat{\Phi}_{v,D(u)} + \hat{\Phi}_{w,D(u)} = \hat{\Phi}_{u,D(u)}$$
3. u **is a product gate:** Assume that u is a product gate in Φ with children v, w. For every $D \in N_u$ and $a \in \{0,1\}^r$ such that $a \leq D(u)$ denote by $D_{v,a} \in N_v$ the function that agrees with D on $\text{path}(u)$ and satisfies $D_{v,a}(v) = a$, and in the same way denote by

$D_{w,a} \in N_w$ the function that agrees with D on path(u) and satisfies $D_{w,a}(w) = a$. The node (u, D) will compute $\hat{\Psi}_{(u,D)}$ from the outputs of the nodes $\{(v, D_{v,a}) : a \leq D(u)\}$ and $\{(w, D_{w,a}) : a \leq D(u)\}$ by the formula

$$\hat{\Psi}_{(u,D)} = \sum_{a \leq D(u)} \hat{\Psi}_{(v,D_{v,a})} \cdot \hat{\Psi}_{(w,D_{w,D(u)-a})}$$

By the induction hypothesis and Equation 4 we have

$$\hat{\Psi}_{(u,D)} = \sum_{a \leq D(u)} \hat{\Psi}_{(v,D_{v,a})} \cdot \hat{\Psi}_{(w,D_{w,D(u)-a})} =$$

$$\sum_{a \leq D(u)} \hat{\Phi}_{v,D_{v,a}(v)} \cdot \hat{\Phi}_{w,D_{w,D(u)-a}(w)} = \sum_{a \leq D(u)} \hat{\Phi}_{v,a} \cdot \hat{\Phi}_{w,D(u)-a} = \hat{\Phi}_{u,D(u)}$$

We fix the output node of Ψ to be the node (u, D) such that u is the output node of Φ and $D \in N_u$ is the function that satisfies $D(u) = \vec{1}$.

Comment: Note that in the construction above there may be nodes (u, D) that are not connected by a path to the output node of Ψ. These nodes do not contribute to the functionality of Ψ and should be removed so that the final Ψ is a tree rather than a union of trees.

Functionality of Ψ:

We proved by induction that every node (u, D) in Ψ computes the polynomial $\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)}$. In particular, the output node (u, D) computes the polynomial $\hat{\Psi}_{(u,D)} = \hat{\Phi}_{u,D(u)} = \hat{\Phi}_{u,\vec{1}} = \hat{\Phi}_u$, which is the polynomial computed by Φ.

Properties of Ψ:

To see that Ψ is a formula note that the output of a node (v, D') is only used by a node (u, D) such that u is the parent of v and D agrees with D' on path(u), and there is at most one such node (u, D). Thus, the out-degree of every node is at most 1.

Ψ is set-multilinear since (by induction) each of its nodes computes a set-multilinear polynomial.

The product-depth of Ψ is the same as the product-depth of Φ since the "product-depth" of Equation 3 is 0 and the "product-depth" of Equation 4 is 1.

Finally, since the size of every N_u is bounded by $(d + 2)^r$, the size of Ψ is at most $O\left((d + 2)^r \cdot s\right)$.

(Note also that we can replace every gate of fanin larger than 2 by a tree of gates of fanin 2, so that the final fanin of every gate in the formula is at most 2). □

Corollary 4. *Let f be a set-multilinear polynomial over sets $X_1, \ldots, X_r$ of size n each, where $r \leq O(\log n / \log\log n)$. If there exists a polynomial size formula for f then there exists a polynomial size set-multilinear formula for f.*

Proof. Let Φ be a polynomial size formula for f. Without loss of generality Φ is a fanin-2 formula of depth $O(\log n)$. The proof hence follows from Theorem 3. □

4 Tensor-Rank and Formula Size

Recall that given a tensor $A : [n]^r \to \mathbb{F}$ and r sets of variables $X_1, \ldots, X_r$, where $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, we defined the set-multilinear polynomial f_A by

$$f_A(x_{1,1}, \ldots, x_{r,n}) = \sum_{i_1,\ldots,i_r \in [n]} A(i_1, \ldots, i_r) \cdot \prod_{j=1}^{r} x_{j,i_j}$$

In this section, we show that for any tensor $A : [n]^r \to \mathbb{F}$ with $r \leq O(\log n / \log\log n)$, if there exists a polynomial size formula for the polynomial f_A then the tensor-rank of A is not too high.

Theorem 5. *Let $n > 1$. Let $A : [n]^r \to \mathbb{F}$ be a tensor such that $r \leq O(\log n / \log\log n)$. If there exists a formula of size n^c for the polynomial f_A then the tensor-rank of A is at most $n^{r \cdot (1 - 2^{-O(c)})}$.*

Proof. Let Φ be a formula of size n^c for f_A. Without loss of generality Φ is a fanin-2 formula of depth $O(\log(n^c))$. Hence, by Theorem 3 we can assume without loss of generality that Φ is a set-multilinear formula (in the sets $X_1, \ldots, X_r$) of size $n^{O(c)}$.

Formulas in Normal Form:

It will be convenient in this proof to allow a leaf of a formula to be labeled by a product of a field element and an input variable (rather than by only one of them). The polynomial computed by such a leaf is the product that labels it.

We will say that a set-multilinear formula is in a *normal form* if it satisfies the following properties (that can be assumed without loss of generality):

1. The fanin of every product gate in the formula is 2.
2. The sum gates in the formula are collapsed so that a child of a sum gate is never a sum gate.
3. No node of the formula computes the 0 polynomial.

4. Every leaf of the formula is labeled by a product of a field element (different than 0) and an input variable (and hence the polynomial computed by any node in the formula is of degree larger than 0).

Without loss of generality we assume that the formula Φ is in normal form.

Tensor-Rank:

We will use the following 3 (straightforward) properties of tensor-rank:

1. For any $A' : [n]^{r'} \to \mathbb{F}$, where $r' > 0$,
$$\text{rank}(A') \leq n^{r'-1} \tag{5}$$
2. For any $A_1, \ldots, A_k : [n]^{r'} \to \mathbb{F}$,
$$\text{rank}\left(\sum_{i=1}^{k} A_i\right) \leq \sum_{i=1}^{k} \text{rank}(A_i) \tag{6}$$
3. For any $A_1 : [n]^{r_1} \to \mathbb{F}$ and $A_2 : [n]^{r_2} \to \mathbb{F}$
$$\text{rank}\left(A_1 \otimes A_2\right) \leq \text{rank}(A_1) \cdot \text{rank}(A_2) \tag{7}$$
(where $A_1 \otimes A_2 : [n]^{r_1+r_2} \to \mathbb{F}$ is the tensor-product of A_1 and A_2, defined by $A_1 \otimes A_2(i_1, \ldots, i_{r_1+r_2}) = A_1(i_1, \ldots, i_{r_1}) \cdot A_2(i_{r_1+1}, \ldots, i_{r_1+r_2})$).

Syntactic-Rank:

For a set-multilinear formula Ψ in normal form in the sets of variables $X_1, \ldots, X_r$, we define the *syntactic-rank* of Ψ inductively as follows:

1. If u is a leaf,
$$\text{syn-rank}(\Psi_u) = 1$$
2. If u is a sum gate with children $u_1, \ldots, u_k$,
$$\text{syn-rank}(\Psi_u) = \min\left(n^{r'-1} \;\; ; \;\; \sum_{i=1}^{k} \text{syn-rank}(\Psi_{u_i})\right)$$
where r' is the degree of $\hat{\Psi}_u$ (and note that since Ψ is in normal form, $r' > 0$).
3. If u is a product gate with children u_1, u_2,
$$\text{syn-rank}(\Psi_u) = \text{syn-rank}(\Psi_{u_1}) \cdot \text{syn-rank}(\Psi_{u_2})$$

By Equation 5, Equation 6, and Equation 7, it is straightforward to verify by induction on the formula that the syntactic-rank bounds the tensor-rank in the following sense: if A_Ψ is the tensor such that the formula Ψ computes the polynomial f_{A_Ψ} then $\text{rank}(A_\Psi) \leq \text{syn-rank}(\Psi)$. Hence, in order to bound the tensor-rank of A it's sufficient to bound the syntactic-rank of Φ.

Extended-Formulas:

It will be convenient to extend the definition of a set-multilinear formula in normal form (in the sets of variables $X_1, \ldots, X_r$), so that each node in the formula can be labeled (in addition to its label in the formula) by an additional real number larger or equal to 1, referred to as the *weight* of the node (if the node is not labeled we think of the weight as 1). We refer to such formulas as *extended-formulas* (in the sets of variables $X_1, \ldots, X_r$). (Note that *extended-formula* already means that the formula is set-multilinear in normal form).

Intuitively, one should think of a weight w of a node as indicating that the sub-formula rooted at that node is duplicated w times and all the w copies are summed. Note however that w is a real number and not necessarily integer.

We stress that an extended-formula is not viewed as a computational device and we will not care about its functionality. We define it in order to explore the relations between size and syntactic-rank of a formula. We will next define the size, the syntactic-degree and the syntactic-rank of an extended formula. Note that the size, the syntactic-degree and the syntactic-rank of an extended formula do not depend on the original labels of the leaves (by field elements and input variables). These labels can be ignored and are irrelevant for the rest of the proof.

Size of an Extended-Formula:

For a leaf u in an extended formula, consider the product of the weights of all the nodes on the directed path from u to the root of the formula. We define the *size* of an extended-formula to be the sum over all the leaves, of the product of the weights of all the nodes on the directed path from u to the root of the formula.

Note that if all weights are 1, the size of an extended-formula is the number of leaves in the formula. This definition is different than our original definition for the size of a formula as the number of edges in the formula, but the two notions differ by a factor of at most 2 and hence the difference will not be important.

Note that if in an extended-formula Ψ, u is a node with children $u_1, \ldots, u_k$, then

$$\text{size}(\Psi_u) = \text{weight}(u) \cdot \sum_{i=1}^{k} \text{size}(\Psi_{u_i}) \tag{8}$$

(where $\text{size}(\Psi_u)$ and $\text{size}(\Psi_{u_i})$ denote the sizes of the extended-formulas Ψ_u and Ψ_{u_i}).

Syntactic-Degree of an Extended Formula:

We define the *syntactic-degree* of an extended-formula Ψ inductively as follows:

1. If u is a leaf, $\text{syn-deg}(\Psi_u) = 1$.

2. If u is a sum gate with children $u_1, \ldots, u_k$, syn-deg(Ψ_u) = syn-deg(Ψ_{u_1}) (note that for every i, j, syn-deg(Ψ_{u_i}) = syn-deg(Ψ_{u_j})).

3. If u is a product gate with children u_1, u_2, syn-deg(Ψ_u) = syn-deg(Ψ_{u_1}) + syn-deg(Ψ_{u_2}).

Syntactic-Rank of an Extended Formula:

We extend the definition of syntactic-rank to extended-formulas. We define the *syntactic-rank* of an extended-formula Ψ (in the sets of variables $X_1, \ldots, X_r$) inductively as follows:

1. If u is a leaf,
$$\text{syn-rank}(\Psi_u) = 1 \tag{9}$$

2. If u is a sum gate with children $u_1, \ldots, u_k$,
$$\text{syn-rank}(\Psi_u) = \min\left(n^{r'-1} \;\;;\;\; \text{weight}(u) \cdot \sum_{i=1}^{k} \text{syn-rank}(\Psi_{u_i})\right) \tag{10}$$
where $r' = \text{syn-deg}(\Psi_u)$ (and note that $r' > 0$).

3. If u is a product gate with children u_1, u_2,
$$\text{syn-rank}(\Psi_u) = \min\left(n^{r'-1} \;\;;\;\; \text{weight}(u) \cdot \text{syn-rank}(\Psi_{u_1}) \cdot \text{syn-rank}(\Psi_{u_2})\right) \tag{11}$$
where $r' = \text{syn-deg}(\Psi_u)$ (and note that $r' > 0$).

Note that if all weights are 1 the definition coincides with the original definition of syntactic-rank.

By the definition, for any node u in Ψ, syn-rank$(\Psi_u) \leq n^{r'-1}$, where $r' = \text{syn-deg}(\Psi_u)$. We say that a node u in Ψ is of *full-rank* if syn-rank$(\Psi_u) = n^{r'-1}$.

Size versus Syntactic-Rank of an Extended-Formula:

For a fixed $s = n^{c'}$, where $1 \leq c' \leq \log r - 3$, consider all of the extended-formulas Ψ' (in the sets of variables $X_1, \ldots, X_r$) such that syn-deg$(\Psi') = r$ and size$(\Psi') \leq s$; and let Ψ be an extended-formula with the largest syntactic-rank among all these formulas[4] (if there are many extended-formulas with the same syntactic-rank, take Ψ to be an extended-formula with the smallest size among all these formulas[5]). We will show that without loss of generality the extended-formula Ψ is of a very specific form.

[4]The maximum exists because the set of extended-formulas Ψ' such that syn-deg$(\Psi') = r$ and size$(\Psi') \leq s$ is a compact space, and syn-rank(Ψ') is a continuous function over that domain.

[5]The minimum exists because the set of extended-formulas Ψ' such that syn-deg$(\Psi') = r$ and size$(\Psi') \leq s$ is a compact space, and syn-rank(Ψ'), size(Ψ') are both continuous functions over that domain.

First we claim that without loss of generality we can assume that every sum gate in Ψ is of fanin 1. Let u be a sum gate in Ψ, with children $u_1, \ldots, u_k$ (where $k > 1$). Note that for every i, syn-deg(Ψ_u) = syn-deg(Ψ_{u_i}). Let $i \in \{1, \ldots, k\}$ be such that syn-rank(Ψ_{u_i})/size(Ψ_{u_i}) is maximal, and let $q = \sum_{j=1}^{k}$ size(Ψ_{u_j}). We will multiply the weight of u by q/size(Ψ_{u_i}) and remove all children of u (together with the sub-trees below them) except u_i. By Equation 8, size(Ψ_u) and hence also size(Ψ) didn't change, while by Equation 10 (and since $i \in \{1, \ldots, k\}$ is such that syn-rank(Ψ_{u_i})/size(Ψ_{u_i}) is maximal), syn-rank(Ψ_u) and hence also syn-rank(Ψ) may only increase. Note also that syn-deg(Ψ) didn't change. Thus, we have turned u into a fanin 1 gate without increasing the size of the extended-formula and without decreasing its syntactic-rank. By repeating this process we can assume that every sum gate in Ψ is of fanin 1.

Next we claim that without loss of generality we can assume that there are no sum gates in Ψ. This is obvious because any sum gate u of fanin 1 can be removed by connecting its child directly to its parent (or just removing u if it has no parent) and passing its weight to its child (that is, multiplying the weight of the child of u by weight(u)). Note that this operation doesn't change size(Ψ) or syn-rank(Ψ) (even if the child of u is a leaf). We can hence assume that Ψ has no sum gates.

Thus, Ψ is a binary tree and every non-leaf node in it is a product gate. Since syn-deg$(\Psi) = r$, the number of leaves in Ψ is r.

We will now analyze the possible weights of the nodes of Ψ.

First we claim that for every leaf u, weight$(u) = 1$. This is because for every leaf u, syn-rank$(\Psi_u) = 1$ (regardless of its weight). Therefore, weight$(u) = 1$, since a higher weight increases the size of the extended-formula without increasing its syntactic-rank.

Let u be a non-leaf node with children u_1, u_2. Denote $r_1 =$ syn-deg(Ψ_{u_1}) and $r_2 =$ syn-deg(Ψ_{u_2}). Thus, syn-deg$(\Psi_u) = r_1 + r_2$. We know that syn-rank$(\Psi_{u_1}) \leq n^{r_1 - 1}$ and syn-rank$(\Psi_{u_2}) \leq n^{r_2 - 1}$. We claim the following:

1. If weight$(u) > 1$ then u_1, u_2 are of full-rank. *Proof:* Assume for a contradiction that weight$(u) > 1$, and u_1 is not of full-rank. (In particular, u_1 is not a leaf). Then by multiplying weight(u_1) by $1 + \epsilon$ and dividing weight(u) by $1 + \epsilon$ (for a small enough ϵ) we preserve the syntactic-rank of Ψ_u (by Equation 11) while decreasing its size (by Equation 8). Thus, we preserve the syntactic-rank of Ψ while decreasing its size, in contradiction to the definition of Ψ.

2. If u is of full-rank then weight$(u) = n$. *Proof:* Since u is of full-rank, syn-rank$(\Psi_u) = n^{r_1 + r_2 - 1}$. Hence by Equation 11, weight$(u) \geq n$. Thus, by 1, u_1, u_2 are of full-rank. Thus, syn-rank$(\Psi_{u_1}) = n^{r_1 - 1}$ and syn-rank$(\Psi_{u_2}) = n^{r_2 - 1}$. Hence, by Equation 11, weight$(u) = n$ (otherwise, by reducing weight(u) to n we preserve the syntactic-rank of Ψ while decreasing its size, in contradiction to the definition of Ψ).

3. By the previous two items, if u is of full-rank then both u_1, u_2 are of full-rank.

Consider the nodes along a path from a leaf to the root in Ψ. The leaf is always of full-rank and, as we saw, the weight of the leaf is 1. By 1,2,3, following the leaf we have a

certain number of full-rank nodes with weight n. After that we have at most one node that is not of full-rank and its weight is larger than 1, and after that we have nodes that are not of full-rank and are with weight 1. Since the size of Ψ is $s = n^{c'}$, the number of full-rank nodes with weight n along a path from a leaf to the root in Ψ is at most $\log_n(s) = c'$.

Denote by V the set of non-leaf nodes in Ψ that are not of full-rank and their weight is 1. Thus, from any leaf of Ψ we can reach a node in the set V by a path of length at most $c' + 2$. Hence, since the number of leaves in Ψ is r and since Ψ is a binary tree, $|V| \geq \frac{r}{2^{c'+2}}$. Also, by 1,2,3, V is a binary tree (of product gates) such that if $v \in V$ and v is a child of u then $u \in V$.

Denote by U the set of nodes that are not in V but are connected to V by an edge, that is, U is the set of direct descendants of V out of V. Let $l = |U|$ and let $u_1, \ldots, u_l$ be the nodes in U and let $r_i = \text{syn-deg}(\Psi_{u_i})$. Note that $l > |V|$ and that for every i, $\text{syn-rank}(\Psi_{u_i}) \leq n^{r_i - 1}$. Since V is a binary tree of product gates $\sum_{i=1}^{l} r_i = \text{syn-deg}(\Psi) = r$, and since the weight of every node in V is 1, by Equation 11 (applied $|V|$ times)

$$\text{syn-rank}(\Psi) \leq \prod_{i=1}^{l} \text{syn-rank}(\Psi_{u_i}) \leq n^{(\sum_{i=1}^{l} r_i) - l} \leq n^{r - r/2^{c'+2}}$$

Since Ψ was chosen to be the extended-formula with the highest syntactic-rank among all the extended-formulas of size $n^{c'}$, we have the same bound for every extended-formula (as a function of its size). In particular, since Φ is a formula of size $n^{O(c)}$,

$$\text{rank}(A) \leq \text{syn-rank}(\Phi) \leq n^{r - r/2^{O(c)}}$$

□

Corollary 6. *Let $A : [n]^r \to \mathbb{F}$ be a tensor such that $r \leq O(\log n / \log\log n)$. If the tensor-rank of A is $\geq n^{r \cdot (1 - o(1))}$ then there is no polynomial size formula for the polynomial f_A.*

(Note that the corollary is interesting only when $r = r(n)$ is super-constant. For $r = O(1)$, the corollary holds trivially since the tensor-rank of any $A : [n]^r \to \mathbb{F}$ is at most n^{r-1}, so the tensor-rank of A is never $\geq n^{r \cdot (1 - o(1))}$).

References

[Aar04] S. Aaronson. Multilinear Formulas and Skepticism of Quantum Computing. STOC 2004: 118-127

[AFT11] B. Alexeev, M. A. Forbes, J. Tsimerman. Tensor Rank: Some Lower and Upper Bounds. Conference on Computational Complexity 2011: 283-291

[AV08] M. Agrawal, V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. FOCS 2008: 67-75

[BS83] W. Baur, V. Strassen. The Complexity of Partial Derivatives. *Theor. Comput. Sci.* 22: 317-330 (1983)

[Gat88] J. von zur Gathen. Algebraic Complexity Theory. *Ann. Rev. Computer Science* 3: 317-347 (1988)

[GK98] D. Grigoriev, M. Karpinski. An Exponential Lower Bound for Depth 3 Arithmetic Circuits. STOC 1998: 577-582

[GR98] D. Grigoriev, A. A. Razborov. Exponential 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) (preliminary version in FOCS 1998)

[H89] J. Håstad. Tensor Rank is NP-Complete. *J. Algorithms* 11(4): 644-654 (1990) (preliminary version in ICALP 1989)

[HY09] P. Hrubeš, A. Yehudayoff. Homogeneous Formulas and Symmetric Polynomials, *Computational Complexity* 20(3): 559-578 (2011)

[Nis91] N. Nisan. Lower Bounds for Non-Commutative Computation. STOC 1991: 410-418

[NW95] N. Nisan, A. Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity* 6(3): 217-234 (1996) (preliminary version in FOCS 1995)

[R04a] R. Raz. Multi-Linear Formulas for Permanent and Determinant are of Super-Polynomial Size. *J. ACM 56(2)* (2009) (Preliminary version in STOC 2004)

[R04b] R. Raz. Separation of Multilinear Circuit and Formula Size. *Theory Of Computing* 2(6) (2006) (preliminary version in FOCS 2004, title: Multilinear-$NC_1 \neq$ Multilinear-NC_2)

[R08] R. Raz. Elusive Functions and Lower Bounds for Arithmetic Circuits. *Theory of Computing* 6(1): 135-177 (2010) (preliminary version in STOC 2008)

[RSY07] R. Raz, A. Shpilka, A. Yehudayoff. A Lower Bound for the Size of Syntactically Multilinear Arithmetic Circuits. *SIAM J. Comput.* 38(4): 1624-1647 (2008) (preliminary version in FOCS 2007)

[RY08a] R. Raz, A. Yehudayoff. Multilinear Formulas, Maximal-Partition Discrepancy and Mixed-Sources Extractors. *J. Comput. Syst. Sci.* 77(1): 167-190 (2011) (preliminary version in FOCS 2008)

[RY08b] R. Raz, A. Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity* 18(2): 171-207 (2009) (preliminary version in CCC 2008)

[Str73] V. Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.* 264: 182-202 (1973)

[SY10] A. Shpilka, A. Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science* 5(3-4): 207-388 (2010)

[Val79] L. G. Valiant. Completeness Classes in Algebra STOC 1979: 249-261

[VSBR83] L. G. Valiant, S. Skyum, S. Berkowitz, C. Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.* 12(4): 641-644 (1983)