Arithmetic circuits: the chasm at depth four gets wider

Pascal Koiran

Arithmetic Circuits: The Chasm at Depth Four Gets Wider

Pascal Koiran
LIP*, École Normale Supérieure de Lyon, Université de Lyon†
Pascal.Koiran@ens-lyon.fr

March 22, 2012

Abstract

In their paper on the "chasm at depth four", Agrawal and Vinay have shown that polynomials in m variables of degree $O(m)$ which admit arithmetic circuits of size $2^{o(m)}$ also admit arithmetic circuits of depth four and size $2^{o(m)}$. This theorem shows that for problems such as arithmetic circuit lower bounds or black-box derandomization of identity testing, the case of depth four circuits is in a certain sense the general case.

In this paper we show that smaller depth four circuits can be obtained if we start from polynomial size arithmetic circuits. For instance, we show that if the permanent of $n \times n$ matrices has circuits of size polynomial in n, then it also has depth 4 circuits of size $n^{O(\sqrt{n}\log n)}$. If the original circuit uses only integer constants of polynomial size, then the same is true of the resulting depth four circuit. These results have potential applications to lower bounds and deterministic identity testing, in particular for sums of products of sparse univariate polynomials. We also use our techniques to reprove two results on:

- The existence of nontrivial boolean circuits of constant depth for languages in LOGCFL.
- Reduction to polylogarithmic depth for arithmetic circuits of polynomial size and polynomially bounded degree.

*UMR 5668 ENS Lyon, CNRS, UCBL, INRIA.

†This work was done during a visit to the Fields Institute and to the University of Toronto's Department of Computer Science.

1 Introduction

Agrawal and Vinay have shown that polynomials of degree $d = O(m)$ in m variables which admit nontrivial arithmetic circuits also admit nontrivial arithmetic circuits of depth four [1]. Here, "nontrivial" means of size $2^{o(d+d\log\frac{m}{d})}$. The resulting depth 4 circuits are $\sum\prod\sum\prod$ arithmetic formulas: the output gate (at depth 4) and the gates at depth 2 are addition gates, and the other gates are multiplication gates. This theorem shows that for problems such as arithmetic circuit lower bounds or black-box derandomization of identity testing, the case of depth four circuits is in a certain sense the general case.

But what if we start from arithmetic circuits of size smaller than $2^{o(m)}$ (for instance, of size polynomial in m)? It is reasonable to expect that the size of the corresponding depth four circuits will be reduced accordingly, but such a result cannot be found in [1]. One of the main results of this paper is a depth reduction theorem for VP families (i.e., families (f_n) of polynomials of degree and arithmetic circuit complexity polynomially bounded in n). We show in Theorem 5 that any VP family (f_n) has depth 4 arithmetic formulas of size $n^{O(\sqrt{d_n}\log d_n)}$, where d_n is the degree of f_n. For instance, this result shows that if the permanent of $n \times n$ matrices has circuits of size polynomial in n, then it also has depth 4 formulas of size $n^{O(\sqrt{n}\log n)}$. This is potentially useful for a lower bound proof: to show that the permanent does not have polynomial size circuits, we "only" have to show that it does not have depth 4 formulas of size $n^{O(\sqrt{n}\log n)}$. This is still certainly far away from the known lower bounds for constant depth arithmetic circuits: currently we have superpolynomial lower bound for the permanent for circuits of depth 3 only, and only in finite fields [5, 6]. In the restricted setting of multilinear arithmetic circuits, superpolynomial lower bounds can be obtained for circuits of arbitrary constant depth [17]. We do not address the issue of multilinearity in this paper. Note however that the results in [16, 17] suggest that the bound in Theorem 5 could be fairly close to optimal at least for multilinear circuits. Indeed, a polynomial f of degree $3n-1$ in $O(n^3)$ variables with multilinear arithmetic circuits of polynomial size is constructed in Section 4 of [16]. By Theorem 4.3 of [16] and Theorem 5.1 of [17], all multilinear depth 4 circuits for f are of size at least $n^{\Omega(\sqrt{n/\log(n)})}$. This shows that the exponent $\sqrt{d_n}$ in Theorem 5 cannot be removed if we insist on a reduction to depth 4 that would preserve multilinearity. Note that for reduction to depth $\log^2(n)$, preservation of multilinearity is indeed possible [16].

We also perform an analysis of the size of the integer constants used

by the depth 4 circuit simulating a given polynomial size circuit (a similar analysis for the construction in [1] has not been carried out yet to the author's knowledge). Roughly speaking, we show that reduction to depth 4 does not require the introduction of large constants. In particular, we give in Theorem 6 an analogue of Theorem 5 for VP^0 (this is a constant-free version of VP). This result is used in [10], where we show that black-box derandomization of identity testing for sums of products of sparse univariate polynomials with sparse coefficients would imply a lower bound for the permanent. Finally, we give applications of our depth reduction techniques to boolean circuit complexity and to the construction of arithmetic circuits of polylogarithmic depth.

1.1 Main Ideas and Comparison with Previous Work

The main depth reduction result in [1] is as follows.

Theorem 1 *Let $P(x_1, \ldots, x_m)$ be a polynomial of degree $d = O(m)$ over a field F. If there exists an arithmetic circuit of size $2^{o(d+d\log\frac{m}{d})}$ for P then there exists a depth 4 arithmetic circuit of size $2^{o(d+d\log\frac{m}{d})}$.*

Theorem 2.4 in [1] also provides some bounds on the fan-in of the gates in the resulting depth 4 circuits.

For multilinear polynomials, their result (Corollary 2.5 in [1]) reads as follows:

Corollary 1 *A multilinear polynomial in m variables which has an arithmetic circuit of size $2^{o(m)}$ also has a depth 4 arithmetic circuit of size $2^{o(m)}$.*

We give the (simple) proof, which is omitted from [1]. For $d = m$ the result is clear since the exponent $d + d\log\frac{m}{d}$ in Theorem 1 is equal to m. Consider now the case of a polynomial $P(X_1, \ldots, X_m)$ of degree $d < m$, having a circuit of size $2^{o(m)}$. Let $Q = P + \prod_{i=1}^{m} X_i$. Since the number of variables of Q is equal to its degree, we are back to the first case: Q has a depth four circuit of size $2^{o(m)}$. We can obtain a circuit of size $2^{o(m)}$ for P by subtracting the product $\prod_{i=1}^{m} X_i$ (this requires only m additional arithmetic operations). Note that this corollary and its proof hold more generally for any (possibly not multilinear) polynomial of degree $d \leq m$.

By specializing the multilinear polynomial to the permanent, Agrawal and Vinay then state in Corollary 2.6 that if every depth 4 arithmetic circuit for the permanent requires exponential size, the same is true for arithmetic circuits of unbounded depth. It is not made precise in [1] what "exponential

size" exactly means. In this context (arithmetic complexity of the permanent) the most standard interpretation is probably that an exponential size circuit for the $n \times n$ permanent is of size $2^{\Omega(n)}$ (note that the number of variables is $m = n^2$). With this interpretation, it is not clear why Corollary 2.6 of [1] would follow from Theorem 1 or Corollary 1.

Since the permanent of a $n \times n$ matrix has degree $d = n$ and $m = n^2$ variables, we can deduce the following from Theorem 1: If there exists an arithmetic circuit of size $2^{o(n \log n)}$ for the $n \times n$ permanent then there exists also a depth 4 arithmetic circuit of size $2^{o(n \log n)}$. This statement is not very useful since we already know (by Ryser's formula [18]) that the permanent has depth 3 arithmetic formulas of size $O(n2^n)$. Note that applying Corollary 1 directly to the permanent would give an even worse bound (namely, we would obtain depth 4 formulas of size $2^{o(n^2)}$). As explained earlier, we can show that if the permanent has polynomial size circuits it must also have depth 4 formulas of size $n^{O(\sqrt{n} \log n)}$. This result does not follow from Theorem 1. On the other hand, our results are weaker than Theorem 1 if we start from a very large circuit. Indeed, as explained below, we can only show that a circuit of size t and degree d has an equivalent depth 4 circuit of size $t^{O(\sqrt{d} \log d)}$. This does not imply Theorem 1.

Before describing their general depth reduction algorithm, Agrawal and Vinay begin with the special case of matrix powering. For this problem there is a very simple and elegant reduction to depth four. Then they treat the general case with an apparently different approach: their construction builds on the depth reduction algorithm of Allender, Jiao, Mahajan and Vinay [3], who gave a uniform version of the depth reduction result due to Valiant, Skyum, Berkowitz and Rackoff [23]. In this paper we show that the matrix powering idea is powerful enough to handle arbitrary polynomial-size arithmetic circuits. Arithmetic branching programs and weakly skew circuits are the main tools that we use to reduce the evaluation of arbitrary arithmetic circuits to matrix powering. These models are known to capture the complexity of a number a problems from linear algebra such as e.g. matrix powering, iterated matrix multiplication or computation of the determinant [19, 12].

1.2 Organization of the paper

In Section 2 we present the two main computation models that we will use: arithmetic circuits and arithmetic branching programs. We define some of the corresponding complexity classes, and give some basic properties. In Section 3, building on a construction of Malod and Portier [12] we give an efficient simulation of arithmetic circuits by arithmetic branching programs.

Compared to [12], we take extra care to construct branching programs of small depth because the square root of the depth appears in the exponent of the size estimate for the final depth 4 circuit. In section 4 we reduce branching programs to depth 4 circuits using the matrix powering idea from [1]. Then we state our main technical result in Theorem 3. We show in particular that an arithmetic circuit of size t and formal degree d has a depth 4 circuit of size $t^{O(\sqrt{d}\log d)}$. We draw some consequences for depth reduction of VP families in Section 5, and for depth reduction of VP^0 families in Section 6.

In Section 7 we give an application of these techniques to boolean circuit complexity. Namely, we show that languages in LOGCFL have constant-depth boolean circuits of size 2^{n^ϵ} (and we briefly present the history of this result).

Finally, we show in Section 8 that the same tools can be used to give a very simple (but suboptimal) proof of the fact that for circuits of polynomially bounded size and degree, reduction to polylogarithmic depth can be achieved while preserving polynomial size [23].

2 Arithmetic Circuits and Branching Programs

We recall that an arithmetic circuit contains addition and multiplication gates. In addition to these arithmetic gates there are input gates, labelled by variables or constants from some field K. An output gate is of fan-out zero. We often assume that there is a single ouptut gate. In this case an arithmetic circuit therefore represents a polynomial with coefficients in K. Without loss of generality, we can and will assume that every input gate has fan-out at most 1 (several input gates can be labeled with the same variable or constant if necessary).

We often assume that the arithmetic gates have arity 2, but in constant-depth circuits we naturally allow addition and multiplication gates of unbounded fan-in (we often also some explicit upper bounds on the fan-in, see for instance Theorem 3). In some of our intermediate constructions (e.g. Proposition 2) we also work with weighted addition gates.

Definition 1 *A n-ary weighted addition gate computes a linear combination $a_1x_1 + \cdots + a_nx_n$ of its inputs $x_1, \ldots, x_n$. Here a_i is the weight associated to the i-th input of the gate. The total weight of the gate is $\sum_{i=1}^{n} |a_i|$.*

For instance, a subtraction gate is a binary weighted addition gate with weights $(1, -1)$. We sometimes refer to binary unweighted addition gates as "ordinary addition gates". The size of a circuit is its total number of gates (including input gates).

Definition 2 *Fix a field K. A sequence (f_n) of polynomials with coefficients in K belongs to* VP *if there exists a polynomial $p(n)$ and a sequence (C_n) of arithmetic circuits such that $\deg(f_n) \leq p(n)$, C_n computes f_n and is of size at most $p(n)$.*

The size constraint implies in particular that f_n depends on polynomially many variables. The above definition is fairly robust. For instance we obtain the same class with circuits using gates of fan-in 2 or of unbounded fan-in, weighted or unweighted addition gates.

An arithmetic formula is a circuit where all gates are of fan-out one, except of course the output gate. In the constant depth setting, arithmetic formulas and arithmetic circuits are polynomially related ([17], Claim 2.2).

The complexity of several problems from linear algebra such as iterated matrix multiplication or computing the determinant is captured by a restricted class of arithmetic circuits called *weakly skew circuits* [19, 12]. Let C be an arithmetic circuit where all multiplication gates are binary. A multiplication gate α in C is said to be disjoint if at least one of its two subcircuits is disjoint from the remainder of C, except of course for the edge from the subcircuit to α (removing this edge would therefore disconnect C). The circuit is weakly skew if its multiplication gates are all disjoint. This definition is usually given only for circuits where all addition gates are binary unweighted, but we will use our slightly more general definition instead (see Propositions 2 and 3).

There is also a closely related notion of *skew circuits* [19, 8, 9]: a circuit with binary multiplication gates is skew if for every multiplication gate at least one of the two incoming edges comes from an input of the circuit. Since we have assumed that that input gates have fan-out at most 1, every skew circuit is also weakly skew.

A circuit where the only constants are from the set $\{0, -1, 1\}$ is said to be constant-free. A constant-free circuit represents a polynomial in $\mathbb{Z}[X_1, \ldots, X_n]$, where $X_1, \ldots, X_n$ are the variables labelling the input gates.

The constant-free model was systematically studied by Malod [11]. In particular, he defined a class VP^0 of polynomial families that are "easy to compute" by constant-free arithmetic circuits. First we need to recall the notion of formal degree:

(i) The formal degree of an input gate is equal to 1.

(ii) The formal degree of an addition gate is the maximum of the formal degrees of its incoming gates, and the formal degree of a multiplication gate is the sum of these formal degrees.

Finally, the formal degree of a circuit is equal to the formal degree of its output gate. This is obviously an upper bound on the degree of the polynomial computed by the circuit. Note that this definition can be applied to circuits with weighted addition gates of arbitrary fan-in. For instance, the polynomial $x - 2y$ can be computed by a circuit containing one ordinary addition gate, one multiplication gate and three inputs labeled by x, y and the constant -2. This circuit has formal degree two. The same polynomial can be computed by another circuit containing a binary weighted adition gate (of total weight $1 + |-2| = 3$) with inputs x and y. The second circuit has formal degree 1.

Definition 3 *A sequence (f_n) of polynomials belongs to VP^0 if there exists a polynomial $p(n)$ and a sequence (C_n) of constant-free arithmetic circuits (with unweighted addition gates) such that C_n computes f_n and is of size and formal degree at most $p(n)$.*

The constraint on the formal degree forbids the computation of polynomials of high degree such as e.g. X^{2^n}; it also forbids the computation of large constants such as 2^{2^n}. The class VP^0 is therefore a strict subset of VP (over the field of rational numbers, or more generally any field of characteristic 0). As for VP we obtain the same class with gates of fan-in 2 or of unbounded fan-in, but of course we cannot allow addition gates with arbitrary weights. We can however allow subtraction gates:

Proposition 1 *Let C be a constant-free circuit of size t and formal degree d, where the arithmetic gates are multiplication, unweighted addition or subtraction gates (all of fan-in 2).*

There is an equivalent constant-free circuit C' of formal degree $d+1$ and size at most $6t+3$, where the arithmetic gates are binary multiplications or ordinary additions.

Proof. We need to get rid of subtraction gates. A first idea would be to write each subtraction $x - y$ as $x + (-1) \times y$, but the cumulative effect of the multiplications $(-1) \times y$ could lead to an increase in the formal degree by more than 1. Instead we will represent each gate α in C by a pair of gates (α_1, α_2) in C'. The output of α will be equal to the differences of the outputs of α_1 and α_2. An input x in C can be represented by the pair $(x, 0)$. To simulate the arithmetic operations in C we use the following rules: $(\alpha_1 - \alpha_2) + (\beta_1 - \beta_2) = (\alpha_1 + \beta_1) - (\alpha_2 + \beta_2)$; $(\alpha_1 - \alpha_2) - (\beta_1 - \beta_2) = (\alpha_1 + \beta_2) - (\alpha_2 + \beta_1)$; $(\alpha_1 - \alpha_2) \times (\beta_1 - \beta_2) = (\alpha_1 \times \beta_1 + \alpha_2 \times \beta_2) - (\alpha_2 \times \beta_1 + \alpha_1 \times \beta_2)$. A straightforward induction shows that the gates in a pair (α_1, α_2) will have

same formal degree as the gate α that they represent. Finally, to complete the construction of C' we come back to our first idea: if (α_1, α_2) is the pair representing the output gate of C, we write the difference $\alpha_1 - \alpha_2$ as $\alpha_1 + (-1) \times \alpha_2$. This increases the formal degree by 1. Each arithmetic operation in C is simulated by at most 6 operations in C', and we need 3 additional gates to perform the final subtraction. $\square$

This modest increase in the formal degree cannot be avoided: without subtraction gates there is no better way to compute the polynomial $f(x) = -x$ than by the formula $f(x) = -1 \times x$, which is of formal degree 2.

Finally we define the notion of arithmetic branching program. This is an edge-weighted directed acyclic graph with two distinguished vertices s and t. The output of the branching program is by definition equal to the sum of the weights of all paths from s to t, where the weight of a path is the product of the weights of its edges. In this paper we assume that the edge weights are constants from some field K or variables. Like an arithmetic circuit, a branching program therefore represents a polynomial with coefficients in K. The depth of a branching program is the length (in number of edges) of the longest path from s to t. The term *arithmetic* (or *algebraic*) *branching program* goes back at least to [15, 4] but these objects were used implicitly much earlier, for instance in [21]. Skew circuits, weakly skew circuits and arithmetic branching programs are essentially equivalent models. Indeed, as shown in [9] they simulate each other with only linear overhead (see [8] for the multilinear case).

3 From Circuits to Branching Programs

We first recall Lemma 4 from [12].

Lemma 1 *Let C be a circuit of size t and formal degree d, containing only binary unweighted arithmetic gates. There exists a weakly skew circuit C' of formal degree d and size at most $t^{\log 2d}$ which computes the same polynomial.*

The fact that C' has same formal degree as C is not explicitly stated in [12], but it can be checked that their construction does satisfy this additional property (more on this in the proof of Proposition 2). We would like to apply this construction not to C itself, but to a "normal form" of C containing weighted addition gates. We begin with an easy lemma.

Lemma 2 *Let C be a circuit made only of input gates and (ordinary) addition or subtraction gates. Each gate of C is equivalent to a weighted addition*

gate of total weight at most 2^s, where s is the number of arithmetic gates in C.

Proof. By induction on s. The result is true for $s \leq 1$ since an input gate can be viewed as a unary weighted addition gate of weight 1, and an ordinary addition or subtraction gate as a binary weighted addition gate of weight 2. For $s > 1$, consider an addition or subtraction gate which is an output of C. By induction hypothesis each of the two inputs of the gate computes a function of the form $\sum_{i=1}^{n} a_i x_i$ where $x_1, \ldots, x_n$ are the inputs of C and $\sum_i |a_i| \leq 2^{s-1}$. Therefore the output gate computes a function of the same form with total weight at most 2^s. $\square$

Lemma 3 *Let C be a circuit containing s (weighted) addition gates and m multiplication gates. There is an equivalent circuit C_+ such that:*

(i) C_+ contains at most s addition gates and m multiplication gates.

(ii) Any input to an addition gate is an input of C_+ or the output of a multiplication gate (in other words, the output of an addition gate can be fed only to multiplication gates).

(iii) If all the addition gates of C are ordinary additions or subtractions, the total weight of every addition gate of C_+ is at most 2^s.

(iv) C^+ is of same formal degree as C.

In this lemma and elsewhere in the paper, "equivalent" means that C_+ computes the same polynomial as C.

Proof of Lemma 3. We will keep the same multiplication gates in C_+ as in C. Consider a multiplication gate in C having at least one addition gate γ as an input. We can view γ as the output of a maximal subcircuit which does not contain any internal multiplication gate (the inputs to the subcircuit are therefore inputs of C or multiplication gates). The output of this subcircuit is a linear function of its inputs. We can therefore replace the subcircuit by a single (weighted) addition gate γ'. Moreover, in the case where all the addition gates of C are ordinary additions or subtractions, γ' can be taken of weight at most 2^s by Lemma 2.

We perform this replacement simultaenously for all addition gates of C feeding into a multiplication gate. If the output of C is a multiplication gate, we are done. If the output is an addition gate, we likewise replace its

maximal subcircuit by a weighted addition gate. The resulting circuit C^+ satisfies properties (i), (ii) and (iii).

A straightforward induction shows that every multiplication gate α of C has same formal degree as the corresponding gate in C^+; and that if α has an input γ which is an addition gate, the formal degree of the corresponding gate γ' in C^+ will be equal to that of γ. Hence property (iv) is satisfied as well. $\square$

The same transformation as in Lemma 1 can be applied to C_+ instead of C. The resulting weakly skew circuit contains weighted addition gates.

Proposition 2 *Let C be a circuit of size t and formal degree d where all multiplication gates are binary. There exists a weakly skew circuit C' of degree d and size at most $t^{\log 2d}$ which computes the same polynomial. In C', any input to an addition gate is an input of the circuit or the output of a multiplication gate. Moreover, if all the addition gates of C are ordinary additions or subtractions, the total weight of every addition gate of C' is at most 2^t.*

Proof. We only give a sketch since this is really the same construction as in Lemma 4 of [12]. We briefly explain below why this construction preserves properties (ii), (iii) and (iv) from Lemma 3, and refer to [12] for more details. To achieve weak skewness C' contains multiple copies of each gate of C^+. Moreover, the connection pattern of C^+ is preserved in the following sense. If α' is a copy of a multiplication gate α then its two inputs β' and γ' are copies of the two inputs β and γ of α. Likewise, for any addition gate α of C^+ the inputs of a copy α' will be copies of its inputs, and moreover α' and α will have the same weights ([12] considers only unweighted binary addition gates, but the general case is identical). In particular, α and α' have same total weight and the inputs to α' are inputs of C' or multiplication gates. A straightforward induction shows that every gate of C^+ has same formal degree as its copies in C'. $\square$

Proposition 3 *Let C be a weakly skew circuit of size m and formal degree d, with weights of addition gates coming from some set W. Assume moreover that any input to an addition gate is an input of C or the output of a multiplication gate. There exists an equivalent arithmetic branching program G of size at most $m+1$ and depth at most $3d-1$. The edges of G are labeled by inputs of C or constants from W.*

Proof. The construction is similar to that of ([12], Lemma 5). The main new point is to check the depth bound. Recall from Section 2 that for every multiplication gate α in C we have an *independent subcircuit* which is connected

to the remainder of C only by the arrow from the subcircuit to α. As in [12] we say that a gate is reusable if it does not belong to any independent subcircuit. Also as in [12], we will prove a version of Proposition 3 for circuits with multiple outputs.

We will show by induction that for any reusable gate α of C there is a vertex t_α in G such that the weight of (s, t_α) is the polynomial computed by α. As to the depth, we will show that if α is an addition gate computing a polynomial of formal degree d_α, the depth of t_α in G (the length of the longest path from s to t_α) is at most $3d_\alpha - 1$; if α is a multiplication or input gate, its depth is at most $3d_\alpha - 2$.

The beginning of the induction is clear: a weakly skew circuit C of size $m = 1$ is reduced to a single gate α labeled by some input x. The corresponding graph G has two nodes s and t, with an edge from s to t labeled by x. We take of course $t_\alpha = t$. We have $d_\alpha = 1$, and this gate is indeed at depth $3d_\alpha - 2 = 1$.

Consider now a weakly skew circuit C of size $m \geq 2$, and let α be one of its ouptut gates. Removing α from C, we obtain a circuit C' of size $m - 1$. By induction hypothesis, there is a corresponding graph G' of size at most m with a distinguished vertex s.

If α is an input gate labeled by x, we obtain G by adding a vertex t_α to G', and an edge from s to t_α labeled by x.

Assume now that α is a (weighted) addition gate, with k (distinct) inputs $\alpha_1, \ldots, \alpha_k$. These k gates must be reusable, so by induction hypothesis we have vertices t_{α_i} in C' so that the weight of (s, t_{α_i}) is equal to the polynomial computed by α_i. Moreover, since α is an addition gate the α_i are multiplication or input gates, and are therefore at depth at most $3d_{\alpha_i} - 2 \leq 3d_\alpha - 2$. We obtain G by adding a new vertex t_α to G', and k new edges from the t_{α_i} to t_α (labeled by the same weights as the incoming edges of the addition gate α). The weight of (s, t_α) in G is clearly equal to the polynomial computed by α, and t_α is at depth at most $(3d_\alpha - 2) + 1 = 3d_\alpha - 1$.

Assume finally that α is a multiplication gate with inputs β and γ. Let C_β and C_γ be the corresponding subcircuits. Since C is weakly skew, one of the two subcircuits (say, C_γ) is independent from the rest of C. Hence $m = m_\beta + m_\gamma + 1$ where m_β and m_γ are the sizes of C_β and C_γ. We can apply separately the induction hypothesis to C_β and C_γ. This yields two graphs G_β and G_γ of respective sizes at most $m_\beta + 1$ and $m_\gamma + 1$, with sources s_β and s_γ. In these graphs there are vertices t_β and t_γ such that the weight of (s_β, t_β) in C_β is equal to the polynomial computed by gate β, and the weight of (s_γ, t_γ) in C_γ is equal to the polynomial computed by gate γ. We construct G from these two graphs by identifying t_β and s_γ. The source of G

is $s = s_\beta$. This graph is of size at most $(m_\beta + 1) + (m_\gamma + 1) - 1 = m \leq m + 1$. In G, the vertex associated to gate α will be $t_\alpha = t_\gamma$. The weight of (s, t_γ) in G is indeed equal to the polynomial computed by gate α. For vertices v in G_γ the weight of (s, v) in G is *not* equal to the weight of (s_γ, v) in G_γ, but as pointed out in [12] this does not matter since these vertices correspond to non-reusable gates of C.

Let d, d_β and d_γ be the formal degrees of the circuits C, C_β and C_γ. By induction hypothesis, t_γ is at depth at most $3d_\gamma - 1$ in G_γ, and t_β is at depth at most $3d_\beta - 1$ in G_β. In G, t_γ is therefore at depth at most $(3d_\beta - 1) + (3d_\gamma - 1) = 3d - 2$. □

Combining Propositions 2 and 3 yields the following result.

Theorem 2 *Let C be a circuit of size t and formal degree d where all multiplication gates are binary. There is an equivalent arithmetic branching program G of size at most $t^{\log 2d} + 1$ and depth at most $3d - 1$. The edges of G are labeled by inputs of C or by constants. Moreover, if all the addition gates of C are ordinary additions or subtractions then these constants are integers of absolute value at most 2^t.*

4 From Branching Programs to Depth-4 Circuits

In this section we complete the reduction to circuits of depth 4.

Lemma 4 *Let G be an arithmetic branching program of size m and depth δ, with edges labeled by elements from some set S. There is an $m \times m$ matrix M such that the polynomial computed by G is equal to the entry at row 1 and column m of the matrix power M^p, for any integer $p \geq \delta$. Moreover, the entries of M are in the set $S \cup \{0, 1\}$.*

Proof. Fix a topological ordering of the nodes of G, with the source s labeled 1 and the target t labeled m. We define M as the adjacency matrix of the graph G' obtained from G by adding a loop of weight 1 on vertex t. In other words, $M_{mm} = 1$ and in all other cases M_{ij} is the (possibly null) weight from node i to node j of G. Note that M is upper-diagonal, with all diagonal entries equal to 0 except M_{mm}. It follows from the classical relation between matrix powering and paths in graphs that $(M^p)_{1m}$ is equal to the sum of weights of all st-paths of length exactly p in G'. This is also the sum of weights of all st-paths of length at most p in G, and for $p \geq \delta$ this is the output of the arithmetic branching program. □

Note that for $p \geq \delta$ all entries of M^p except $(M^p)_{1m}$ are equal to zero.

In the last step in our series of reduction, we explain (following basically the same strategy as in [1]) how to perform the matrix powering operation in the above lemma with depth four formulas, and also depth four circuits.

Proposition 4 *Let G be an arithmetic branching program of size m and depth δ. There is an equivalent depth four circuit Γ with m^2+1 unweighted addition gates and $m^{\lceil\sqrt{\delta}\rceil+1}+m^{\lceil\sqrt{\delta}\rceil-1}$ multiplication gates. There is also an equivalent depth four formula Γ_f with $m^{\lceil\sqrt{\delta}\rceil-1}+1$ unweighted addition gates and $m^{\lceil\sqrt{\delta}\rceil-1}+m^{2\lceil\sqrt{\delta}\rceil-2}$ multiplication gates.*

The inputs of Γ and Γ_f are from the set as the edge labels of G, and their multiplication gates are of fan-in $\lceil\sqrt{\delta}\rceil$.

Proof. We need to compute M^p, where $p \geq \delta$ and M is as in Lemma 4. Let p be the smallest square integer bigger or equal to δ. From M we will compute $N = M^{\sqrt{p}}$ by a depth 2 circuit Γ_2, and then from N we will compute $M^p = N^{\sqrt{p}}$ using the same circuit. With a depth 2 circuit one cannot play clever tricks: we can only expand a polynomial as a sum of monomials. In this case we express each entry of N as a sum of $m^{\sqrt{p}-1}$ products of length $\sqrt{p}$, by brute-force expansion of the product $M^{\sqrt{p}}$. This yields a circuit Γ_2 with m^2 addition gates (one for each entry of N) and $m^{\sqrt{p}+1}$ multiplication gates. We can double those estimates to upper bound the size of Γ. To arrive at the slightly better estimate in the statement of Proposition 4, note that the second copy of Γ_2 only needs to compute a single entry of $N^{\sqrt{p}}$.

In order to obtain an arithmetic formula, we recompute from scratch each entry of N whenever it is used by the second copy of Γ_2. The arithmetic formula therefore computes a sum of $m^{\sqrt{p}-1}$ products, where each product is a sum of $m^{\sqrt{p}-1}$ products of entries of M. We therefore have one addition and $m^{\sqrt{p}-1}$ products gates in the top two levels, $m^{\sqrt{p}-1}$ addition and $m^{2(\sqrt{p}-1)}$ multiplication gates in the two bottom levels. □

Note the significant saving in the number of addition gates if we use depth four circuits instead of depth four formulas. We can now prove our main depth reduction result.

Theorem 3 *Let C be an arithmetic circuit of size t and formal degree d where all multiplication gates are binary. There is an equivalent depth four circuit Γ with at most $(t^{\log 2d}+1)^2+1$ unweighted addition gates and at most $2(t^{\log 2d}+1)^{\sqrt{3d}+2}$ multiplication gates.*

There is an equivalent arithmetic formula Γ_f of depth four with at most $(t^{\log 2d}+1)^{\sqrt{3d}}+1$ unweighted addition gates and at most $2(t^{\log 2d}+1)^{2\sqrt{3d}}$

multiplication gates. The inputs of Γ *and* Γ_f *are inputs of* C *or constants; their multiplication gates are of fan-in at most* $\sqrt{3d}+1$.

If C *is constant-free, and if all the addition gates of* C *are ordinary additions or subtractions, then these constants are integers of absolute value at most* 2^t.

Proof. Combine Theorem 2 and Proposition 4. □

Remark 1 *We can obtain smaller circuits for* C *by going for a constant depth larger than four. Let* M *be the matrix in the proof of Proposition 4. To compute a power* M^p *we can start from* M *and raise our matrix to the power* $\sqrt[\Delta]{p}$ *repeatedly (*Δ *times). If we implement each of the* Δ *powerings by a depth 2 circuit, we obtain for the branching program* G *a circuit of depth* 2Δ *and size* $m^{O(\sqrt[\Delta]{p})}$*, for any constant* $\Delta \geq 2$*. For* C*, this translates into a circuit of depth* 2Δ *and size* $t^{O(\sqrt[\Delta]{d}\log d)}$.

If we start from arithmetic formulas (or more generally weakly skew circuits) instead of general arithmetic circuits, we can obtain depth four formulas and circuits of smaller size than in Theorem 3. Indeed, in this case we do not need the transformation from arithmetic circuits to weakly skew circuits given by Proposition 2. This saves a factor of roughly $\log 2d$ in the exponents of our complexity bounds.

Theorem 4 *Let* C *be a weakly skew circuit of size* t *and formal degree* d*. There is an equivalent depth four circuit* Γ *with at most* $(t+1)^2+1$ *unweighted addition gates and at most* $2(t+1)^{\sqrt{3d}+2}$ *multiplication gates.*

There is an equivalent arithmetic formula Γ_f *of depth four with at most* $(t+1)^{\sqrt{3d}}+1$ *unweighted addition gates and at most* $2(t+1)^{2\sqrt{3d}}$ *multiplication gates. The inputs of* Γ *and* Γ_f *are inputs of* C *or constants; their multiplication gates are of fan-in at most* $\sqrt{3d}+1$.

If C *is constant-free, and if all the addition gates of* C *are ordinary additions or subtractions, then these constants are integers of absolute value at most* 2^t.

Proof. Before applying Proposition 3 we make sure that any input to an addition of gate of C is an input of the circuit or a multiplication gate. By Lemma 3 this condition can be ensured without increasing the size of C (and this transformation preserves weak skewness). Hence there is an equivalent arithmetic branching program of size at most $t+1$ and depth at most $3d-1$. Then we convert this branching program into a depth 4 circuit or a depth 4 formula using Proposition 4.

When C is constant free, the bound on the absolute value of the constants of Γ and Γ_f comes (as in Theorem 3) from property (iii) in Lemma 3. $\square$

The savings in the number of addition gates in depth four circuits compared to depth four formulas are especially significant in the above theorem: our circuits contain only quadratically many addition gates. This is a relevant parameter since the number of addition gates (minus 1) is equal to the number of *distinct* sparse polynomials in a sum of products of sparse polynomials [10].

5 Depth Reduction for VP

In accordance with Definition 1, a unary weighted addition gate outputs $\alpha \cdot x$, where α is the weight of the gate and x its input. Recall also from the definition of formal degree in Section 2 that the formal degree of such a gate is equal to that of its input.

The following result is essentially Lemma 2 from [11], written in a different language. We give the proof because we will build on it in the next section.

Proposition 5 *Any* VP *family* (f_n) *can be computed by a polynomial-size family* (C_n) *of circuits of formal degree* $\deg(f_n)$. *The addition gates of* C_n *are unary weighted or binary unweighted (i.e., "ordinary").*

Proof. Since (f_n) is in VP, this family can be computed by a family (C'_n) of arithmetic circuits of polynomial size where all the arithmetic gates are binary unweighted. To construct C_n from C'_n we use a small variation on the standard homogenization trick. In order to homogenize C'_n one would normally represent each gate γ computing a polynomial f_γ by a sequence γ_i of $d_n + 1$ gates, where i ranges from 0 to d_n and γ_i computes the homogenous component of f_γ of degree i. The homogenous components of degree higher than d_n can be discarded since they cannot contribute to the final output. This construction preserves polynomial circuit size, and each gate now computes a polynomial of degree at most d_n. But formal degree can be higher due to multiplication by constants (i.e., homogenous components of degree 0).

To circumvent this difficulty, we get rid of the gates γ_0 representing homogenous components of degree 0. We will therefore construct a circuit C''_n which computes the sum of all homogenous components of f_n of degree at least 1. Our final circuit C_n will then add the output of C''_n to the constant term of f_n, at the cost of one additional arithmetic operation.

We will use unweighted addition gates inside C''_n. Indeed, let γ be a multiplication gate of C_n with inputs α and β. To obtain $f_{\gamma,i}$, the homogenous component of degree i, one normally writes $f_{\gamma,i} = \sum_{j=0}^{i} f_{\alpha,j} f_{\beta,i-j}$. This expression involves $f_{\alpha,0}$ and $f_{\beta,0}$, which as we have said are not represented by any gate of C''_n. Therefore, to compute e.g. $f_{\alpha,0} f_{\beta,i}$, instead of a multiplication gate we use a unary addition gate with input $f_{\beta,i}$ and weight $f_{\alpha,0}$. A straightforward induction shows that a gate γ_i in C''_n will have formal degree i. As a result, C''_n and C_n will be of formal degree d_n. $\square$

Theorem 5 *Let (f_n) be a VP family of polynomials of degree $d_n = \deg(f_n)$. This family can be computed by a family (Γ_n) of depth four circuits with $n^{O(\log d_n)}$ addition gates and $n^{O(\sqrt{d_n} \log d_n)}$ multiplication gates. The family (f_n) can also be computed by a family (F_n) of depth four arithmetic formulas of size $n^{O(\sqrt{d_n} \log d_n)}$. The inputs to Γ_n and F_n are variables of f_n or constants; their multiplication gates are of fan-in at most $\sqrt{3d_n} + 1$.*

Proof. This is an application of Theorem 3: t is polynomial in n, and by Proposition 5 we can take $d = d_n$. $\square$

6 Depth Reduction for VP^0

We first show that a circuit of small size and degree where all inputs are in $\{-1, 0, 1\}$ cannot compute a large integer.

Lemma 5 *Let C be a constant-free and variable-free circuit of size t and formal degree d where all arithmetic gates are binary unweighted. The output of C is an integer of absolute value at most 2^{td}.*

Proof. By induction on t. For $t = 1$ the circuit contains a single input gate, which must carry an integer in $\{-1, 0, 1\}$. The result is therefore true for $t = 1$. Consider now a circuit C of size $t \geq 2$, and let d_1 and d_2 be the formal degrees of the two inputs to the output gate. By induction hypothesis these two gates carry integers of absolute value at most $2^{(t-1)d_1}$ and $2^{(t-1)d_2}$. If the output gate is an addition we have $d_1, d_2 \leq d$ and C therefore computes an integer of absolute value at most $2^{(t-1)d} + 2^{(t-1)d} \leq 2^{td}$. If the output gate is a multiplication, we have $d = d_1 + d_2$ and C computes an integer of absolute value at most $2^{(t-1)d_1} \times 2^{(t-1)d_2} \leq 2^{td}$. $\square$

Proposition 6 *Any VP^0 family (f_n) can be computed by a family (C_n) of constant-free circuits of polynomial size and formal degree $\deg(f_n)$. The arithmetic gates of C_n are binary multiplication, ordinary addition or subtraction gates.*

Proof. Since (f_n) is in VP^0, this family can be computed by a family (C'_n) of constant-free circuits of polynomial size and polynomial formal degree. All the arithmetic gates of C'_n can be assumed to be binary unweighted. To construct C_n from C'_n we proceed along the same lines as in Proposition 5. In particular, we will again construct a circuit C''_n which computes the sum of all homogenous components of f_n of degree at least 1. Our final circuit C_n then adds the output of C''_n to the constant term of f_n (call it c_n). By Lemma 5, c_n has polynomial bit size (it is equal to the output of C'_n when all variables are set to 0). We can therefore compute $|c_n|$ from scratch using a sequence of multiplications by 2 and additions of bits. We use an addition to perform a multiplication by 2, so this construction does not require any multiplication gate. Finally, depending on the sign of c_n we add or subtract $|c_n|$ to the output of C''_n. The resulting circuit C_n will have same formal degree as C''_n.

We also need to use a similar trick inside C''_n. Indeed, let γ be a multiplication gate of C_n with inputs α and β. To obtain $f_{\gamma,i}$, the homogenous component of degree i, one normally writes $f_{\gamma,i} = \sum_{j=0}^{i} f_{\alpha,j} f_{\beta,i-j}$. This expression involves $f_{\alpha,0}$ and $f_{\beta,0}$, which as explained in the proof of Proposition 5 are not represented by any gate of C''_n. Therefore, to compute e.g. $f_{\alpha,0} f_{\beta,i}$ we start from $f_{\beta,i}$ and compute the product using a sequence of multiplications by 2 and additions of $f_{\beta,i}$. As explained above, thanks to Lemma 5 this can be done with a polynomial number of addition gates, at most one subtraction and no multiplication gate. A straightforward induction shows that a gate γ_i in C''_n will have formal degree i. As a result, C''_n and C_n will be of formal degree d_n. □

By Proposition 1, one can get rid of the subtraction gates in Proposition 6 at the cost of a linear increase in circuit size and an increase in the formal degree by just 1 (using Lemma 3 from [11] instead of Proposition 1 would give a worse degree bound).

Theorem 6 *Let (f_n) be a VP^0 family of polynomials of degree $d_n = \deg(f_n)$. This family can be computed by a family (Γ_n) of depth four circuits with $n^{O(\log d_n)}$ addition gates and $n^{O(\sqrt{d_n} \log d_n)}$ multiplication gates. The family (f_n) can also be computed by a family (F_n) of depth four arithmetic formulas of size $n^{O(\sqrt{d_n} \log d_n)}$. The inputs to Γ_n and F_n are variables of f_n or relative integers of polynomial bit size; their multiplication gates are of fan-in at most $\sqrt{3d_n} + 1$.*

Proof. This is an application of Theorem 3: t is polynomial in n, and by Proposition 6 we can take $d = d_n$. □

7 Application to Boolean Circuits

In this section we give an application of our results to boolean circuit complexity. A discussion of depth reduction in the boolean versus arithmetic setting can already be found in [1], but that paper did not actually provide any result of this type. Here we use arithmetic techiques to reprove a known result : languages in LOGCFL have nontrivial constant-depth circuits.

Proposition 7 *Let L be a languange in* LOGCFL*. For every $\epsilon > 0$, L can be decided by a family of constant-depth circuits Γ_n of size 2^{n^ϵ}. The gates of Γ_n are OR or AND gates, both of unbounded fan-in, and NOT gates.*

Proof Sketch. It is known that languages in LOGCFL can be recognized by families (C_n) of semi-unbounded circuits of logarithmic depth and polynomial size [25]. Each circuit C_n has $2n$ inputs; the remaining gates are AND gates of fan-in 2 or OR gates of unbounded fan-in. A language L in LOGCFL is recognized by the corresponding circuit family in the following sense: a word $x \in \{0,1\}^n$ belongs to L iff the input $x_1 \dots x_n \overline{x_1} \dots \overline{x_n}$ is accepted by C_n.

We view C_n as an arithmetic circuit over the boolean semiring $\mathcal{R} = (\{0,1\}, \vee, \wedge)$: the boolean OR is the addition of $\mathcal{R}$, and the boolean AND is its multiplication. The semi-unboundedness property together with the $O(\log n)$ depth bound imply that C_n is of polynomially bounded formal degree. It follows that we can apply the results of Section 4 (up to now we have considered only arithmetic circuits over fields, but the main results and their proofs apply to semirings). The existence of a suitable constant-depth circuit family (Γ_n) therefore follows from Remark 1. Note that the depth of Γ_n depends on the exponent in the polynomial bound for the formal degree of C_n. □

Remark 2 *Instead of working over the semiring $\mathcal{R}$ in the above proof, one could also work over $(\mathbb{N}, +, \times)$. To do this replace each OR gate of C_n by a $+$ gate and each AND gate by a $\times$ gate; apply Remark 1 to the resulting circuit; and finally convert back addition gates into OR gates and multiplication gates into AND gates.*

One can find in Lemma 8.1 of [2] a proof of Proposition 7 for languages in NL (a subset of LOGCFL), and the authors observe that the proof also applies to LOGCFL. According to [26], the result for NL is usually credited to Nepomnjascii [14]. Nepomnjascii proved a uniform version of this result which in recent years has been used in time-space lower bounds (see [24] for

a survey on this topic). The result for languages in L was used in [7] to construct certain uniform families of expanders.

Another depth reduction result due to Valiant shows that boolean circuits of linear size and depth $O(\log n)$ have depth-3 circuits of size $2^{O(n/\log\log n)}$. This result is stated in [22] for monotone circuits. The statement for non-monotone circuits (and a proof based on [20, 22]) can be found in [26]. All these results suggest that lower bounds on the size of circuits of logarithmic depth might be obtained by proving strong enough lower bounds for constant-depth circuits (and quite possibly explain why it is difficult to obtain very strong lower bounds for constant-depth circuits).

8 Reduction to Polylogarithmic Depth

It was shown by Valiant, Skyum, Berkowitz and Rackoff [23] that arithmetic circuits of polynomially bounded size and degree can be transformed into circuits of polylogarithmic depth and polynomial size (the depth can even be made logarithmic with addition gates of unbounded fan-in). Since then several refinements of this fundamental result have been published, adressing in particular the issues of uniformity [13, 3] or multilinearity [16]. In this section we give another proof of reduction to polylogarithmic depth. The depth bound that we obtain is worse than [23] by a logarithmic factor. This result is therefore not new neither optimal, but nonetheless we feel that it is worth presenting here because its proof is quite simple and based on the same tools as the remainder of the paper: (weakly) skew circuits and arithmetic branching programs.

Before turning to general arithmetic circuits, we first parallelize arithmetic branching programs.

Proposition 8 *Let G be a (multi-output) arithmetic branching program of size m and depth δ. There is a multi-output arithmetic circuit C of depth $2\lceil\log\delta\rceil$ which computes the m polynomials represented by the m nodes of G. The circuit contains $m^3\lceil\log\delta\rceil$ binary multiplication gates and $m^2\lceil\log\delta\rceil$ addition gates of unbounded fan-in.*

Proof. It is again based on matrix powering. We start from the adjacency matrix of G, and add the identity matrix (instead of a single 1 on the diagonal as in the proof of Lemma 4). Let M be the resulting matrix. Assuming again that the source node of G is labeled 1, the polynomial represented by node j of G is equal to $(M^p)_{1j}$ for any power $p \geq \delta$. We will compute M^p by repeated squaring. From M we can compute M^2 by a depth 2 circuit

with m^3 multiplication gates and m^2 unbounded additions. We repeat this process $\lceil \log \delta \rceil$ times to obtain M^p. $\square$

Theorem 7 *Let C be a circuit of size t and formal degree d where all multiplication gates are binary. There is an equivalent circuit C' (with binary multiplication gates as well) of depth $O(\log t \cdot \log d)$ and size $O(t^3 \log t \cdot \log d)$*

Proof. We decompose C in "layers" C_i: C_i is made of all gates of C of formal degree in the interval $[2^i, 2^{i+1}[$. Here i ranges from 0 to $\lfloor \log d \rfloor$. Each layer forms a (multi-output) arithmetic circuit; for $i \geq 1$, the input gates of C_i actually belong to previous C_j's for various $j < i$. The crucial observation is that these arithemetic circuits are all skew, i.e., for each mutiplication gate at least one of the two arguments is an input gate of C_i. Indeed, the product of two gates of formal degree at least 2^i is of formal degree at least 2^{i+1} and therefore cannot belong to C_i. But (as pointed out at the end of Section 2) skew circuits and arithmetic branching programs are essentially equivalent objects. In particular, by Lemma 5 of [12] a skew circuit (or even a weakly skew circuit) of size s can be simulated by an arithmetic branching program of size $s+1$ (this result of [12] is stated only for circuits with binary addition gates, but the proof clearly applies to unbounded fan-in as well[1]). By Proposition 8 each C_i is therefore equivalent to a circuit of depth $O(\log t)$ and size $O(t^3 \log t)$. We multiply these estimates by $1 + \lfloor \log d \rfloor$ to obtain the final result. $\square$

Acknowlegments

I thank Eric Allender, Bruno Grenet, Natacha Portier and Amir Yehudayoff for useful discussions on this work.

References

[1] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proc. 49th IEEE Symposium on Foundations of Computer Science*, pages 67–75, 2008.

[2] E. Allender, L. Hellerstein, P. Mccabe, T. Pitassi, and M. Saks. Minimizing DNF formulas and AC^0 circuits given a truth table. To appear in *SIAM Journal on Computing*. Preliminary version in *Proc. 2006 Conference on Computational Complexity.*

[1] We gave in Proposition 3 a variation on this result.

[3] E. Allender, J. Jiao, M. Mahajan, and V. Vinay. Non-commutative arithmetic circuits: depth reduction and size lower bounds. *Theoretical Computer Science*, 209:47–86, 1998.

[4] A. Beimel and A. Gál. On arithmetic branching programs. *Journal of Computer and System Sciences*, 59(2):195–220, 1999.

[5] D. Grigoriev and M. Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *Proc. 30th ACM Symposium on Theory of Computing*, pages 577–582, 1998.

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

[7] D. Gutfreund and E. Viola. Fooling parity tests with parity gates. In *Proc. APPROX and RANDOM 2004*, LNCS 3122, pages 381–392. Springer, 2004.

[8] M. Jansen. Lower bounds for syntactically multilinear algebraic branching programs. In *Proc. MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 407–418. Springer-Verlag, 2008.

[9] E. Kaltofen and P. Koiran. Expressing a fraction of two determinants as a determinant. In *Proc. 2008 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 141–146. ACM Press, 2008.

[10] P. Koiran. Shallow circuits with high-powered inputs. In *Proc. Seoond Symposium on Innovations in Computer Science (ICS 2011)*. Preprint: http://arxiv.org/abs/1004.4960, 2010.

[11] G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard - Lyon 1, 2003.

[12] G. Malod and N. Portier. Characterizing Valiant's complexity classes. *Journal of Complexity*, 24:16–38, 2008. Conference version in MFCS 2006.

[13] G. Miller, V. Ramachandran, and E. Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. In *VLSI Algorithms and Architectures (Proc. Aegean Workshop on Computing)*, LNCS 227. Springer, 1986.

[14] V. Nepomnjascii. Rudimentary predicates and Turing calculations. *Soviet Mathematics Doklady*, 11(6):1462–1465, 1970.

[15] N. Nisan. Lower bounds for non-commutative computation. In *Proc. 23rd ACM Symposium on Theory of Computing*, pages 410–418, 1991.

[16] R. Raz and A. Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008.

[17] R. Raz and A. Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009.

[18] H. J. Ryser. *Combinatorial Mathematics*, volume 201 of *Carus mathematical monographs*. Mathematical Association of America, 1963.

[19] S. Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE T. Inf. Syst.*, 75(1):116–124, 1992.

[20] L. G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 1977.

[21] L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.

[22] L. G. Valiant. Exponential lower bounds for restricted monotone circuits. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 110–116, 1983.

[23] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.

[24] D. van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science*, 2:197–303, 2007.

[25] H. Venkateswaran. Properties that characterize LOGCFL. *Journal of Computer and System Sciences*, 43(2):380–404, 1991.

[26] E. Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.