

SMALL SPACE ANALOGUES OF VALIANT'S CLASSES AND THE LIMITATIONS OF SKEW FORMULAS

MEENA MAHAJAN AND B. V. RAGHAVENDRA RAO

Abstract. In the uniform circuit model of computation, the width of a boolean circuit exactly characterizes the “space” complexity of the computed function. Looking for a similar relationship in Valiant’s algebraic model of computation, we propose width of an arithmetic circuit as a possible measure of space. In the uniform setting, we show that our definition coincides with that of VPSPACE at polynomial width. We introduce the class VL as an algebraic variant of deterministic log-space L ; VL is a subclass of VP .

Further, to define algebraic variants of non-deterministic space-bounded classes, we introduce the notion of “read-once” certificates for arithmetic circuits. We show that polynomial-size algebraic branching programs (an algebraic analog of NL) can be expressed as read-once exponential sums over polynomials in VL , *i.e.* $\text{VBP} \in \Sigma^R \cdot \text{VL}$. Thus, read-once exponential sums can be viewed as a reasonable way of capturing space-bounded non-determinism. We also show that $\Sigma^R \cdot \text{VBP} = \text{VBP}$, *i.e.* VBPs are stable under read-once exponential sums. Though the best upper bound we have for $\Sigma^R \cdot \text{VL}$ itself is VNP , we can obtain better upper bounds for width-bounded multiplicatively disjoint (md-) circuits. Without the width restriction, md- arithmetic circuits are known to capture all of VP . We show that read-once exponential sums over md- constant-width arithmetic circuits are within VP and that read-once exponential sums over md- polylog-width arithmetic circuits are within VQP .

We also show that exponential sums of a skew formula cannot represent the determinant polynomial.

Keywords. Arithmetic circuits, Valiant’s classes, space complexity, circuit width, algebraic branching programs.

Subject classification. 68Q05, 03D15, 68Q15.

1. Introduction

Valiant introduced the classes VP and VNP in algebraic complexity theory to capture the complexity of algebraic computation of polynomial families [Valiant \(1979a, 1982\)](#) (see also [Bürgisser 2000](#)). Over Boolean computation these classes correspond roughly to P and NP; over arithmetic computation with Boolean inputs they correspond roughly to $\#\text{LogCFL}$ ([Vinay 1991](#)) and $\#\text{P}$ ([Valiant 1979b](#)). Given the rich structure within P and LogCFL ([Cook 1971](#)), it is natural to ask for a complexity theory that can describe algebraic computation at this level. In particular, there are two well-known hierarchies within polynomial-size Boolean circuit families: the NC hierarchy based on depth, modeling parallel time on a parallel computer ([Borodin 1977](#), see also [Greenlaw *et al.* 1995](#)), and the SC hierarchy based on width, modeling simultaneous time-space complexity of P machines (see [Johnson 1990](#); [Cook 1979](#); [Pippenger 1979](#)). It is straightforward to adapt Valiant’s definition of VP to classes like AC^0 and NC^1 . But an adaptation capturing a space-bound is more tricky, especially when dealing with sub-linear space.

The main obstacle in this direction is defining a “right” measure for space. Two obvious choices are as follows: 1) the number of arithmetic “cells” or registers used during the course of computation (*i.e.* the unit-space model) and 2) the size of a succinct description of the polynomials computed at each cell. A third choice is the complexity of computing the coefficient function for polynomials in the family. All three of these space measures have been studied in the literature, [Valiant \(1976\)](#), [Nisan & Wigderson \(1995\)](#), [Michaux \(1989\)](#), [de Naurois \(2006\)](#), [Koiran & Perifel \(2009b,a\)](#), with varying degrees of success. As the brief overview in [Section 3.1](#) shows, the models of [Michaux \(1989\)](#), [Koiran & Perifel \(2009b,a\)](#) when adapted to logarithmic space are too powerful to give meaningful insights into small-space classes, whereas the model of [de Naurois \(2006\)](#) as defined for log-space is too weak.

The main purpose of this paper is to propose yet another model for describing space-bounded computations of families of polynomials. Our model is based on the width of arithmetic circuits (see [Limaye *et al.* 2010](#); [Arvind *et al.* 2010](#); [Jansen & Sarma 2010](#) for more on width-bounded arithmetic circuits) and captures both succinctness of coefficients and ease of evaluating the polynomials. We show that our notion of space $\text{VSPACE}(s)$ coincides with that of [Koiran & Perifel \(2009a,b\)](#) at polynomial space with uniformity ([Theorem 3.2](#)), and so far avoids the pitfalls of being too powerful or too weak at logarithmic space VL (see [Lemma 3.9](#)).

Continuing along this approach, we propose a new way of describing non-deterministic space-bounded computation in this context. Valiant's framework captures non-determinism by *defining* VNP as $\Sigma \cdot \text{VP}$. However, for non-deterministic log-space NL , there is a well-known model that directly carries over to the arithmetic setting, namely polynomial-size branching programs BP . (See [Section 2.1](#) for formal definitions.) Thus, treating VBP as VNL , we would like an analogous characterization of the form $(\text{VNL} =) \text{VBP} = \Sigma \cdot \text{VL}$. However, this does not quite work, since $\Sigma \cdot \text{VL}$ equals all of VNP . (This follows from the facts that $\text{VNP} = \Sigma \cdot \text{VNC}^1$, see [Bürgisser \(2000\)](#), and that $\text{VNC}^1 \subseteq \text{VL}$.) Hence, we use another model for non-determinism based on read-once certificates; this also provides the correct description of NL in terms of L in the Boolean world in the sense that $\text{NL} = \exists^R \cdot \text{L}$. We show that the algebraization of this model, $\Sigma^R \cdot \text{VL}$, does contain arithmetic branching programs VBP ([Theorem 4.4](#)).

Surprisingly, we are unable to show the converse. In fact, we are unable to show any good (non-trivial) upper bound on the complexity of read-once certified log-space polynomial families. This raises the question: Is the read-once certification procedure inherently too powerful? We show that this is not always the case; for branching programs, read-once certification adds no power at all ([Theorem 4.8](#)). Further, if the circuit is multiplicatively disjoint and of constant width, then read-once certification does not take us beyond VP ([Theorem 5.1](#)).

We also study the class of polynomial-size skew formulas, denoted SkewF . The motivation for this study arises from Valiant's

characterizations of the classes VP and VNP (see Valiant 1979b). Valiant proved that every polynomial $p(X) \in \text{VNP}$, and in particular every polynomial in VP, can be written as $p(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where the polynomial ϕ has an arithmetic formula, or expression, of polynomial size. We know that the family of permanent polynomials is complete for VNP (see e.g. Bürgisser 2000). It is also known Toda (1991) that the family of determinant polynomials is complete for the class of polynomials computed by skew circuits of polynomial size. The question we ask is: can we prove a similar equivalence in the case of skew circuits? That is, can we write polynomials computed by skew circuits as an exponential sum of polynomials computed by skew formulas? We show that this is not possible, by showing that any polynomial which is expressible as an exponential sum of a skew formula can again be represented by a skew formula.

The rest of the paper is organized as follows: Section 3 gives a detailed account of existing notions of space for algebraic computation and introduces circuit width as a possible measure of space. In Section 4, we introduce the notion of read-once certificates and read-once exponential sums. Section 5 contains upper bounds for read-once exponential sums of some restricted circuit classes. In Section 6, we present the limitations of skew formulas.

2. Preliminaries

We use standard definitions for complexity classes such as polynomial space PSPACE, logarithmic space L, non-deterministic log-space NL (see e.g. Vollmer 1999; Arora & Barak 2009).

2.1. Circuit classes. A Boolean circuit is a directed acyclic graph C , where nodes with non-zero in-degree are labeled from $\{\vee, \wedge, \neg\}$, and nodes of zero-in-degree (called *leaf* nodes) are labeled from $X \cup \{0, 1\}$, where $X = \{x_1, \dots, x_n\}$ is the set of variable inputs to the circuit. An output node of C is a node of zero out-degree, and it computes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Without loss of generality, we can assume that negations appear only at the leaves.

An arithmetic circuit over a ring $\mathbb{K} = \langle K, +, \times, 0, 1 \rangle$ is similarly a directed acyclic graph C , where nodes with non-zero in-degree are labeled from $\{+, \times\}$, and nodes of zero-in-degree (called *leaf* nodes) are labeled from $X \cup K$. An output node of C computes a polynomial in $\mathbb{K}[X]$. (A circuit can have more than one output node, thus computing a set of polynomials.)

The following definitions apply to both arithmetic and boolean circuits, hence we simply use the term circuit. The *depth* of a circuit is the length of a longest path from a leaf node to an output node, and its *size* is the number of nodes and edges in it. The circuit is said to be *layered* if the vertices can be partitioned into sets V_0, V_1, \dots, V_d such for every edge e there is an i such that e is directed from V_i to V_{i+1} . The sets V_0, V_1, \dots, V_d are the layers of the circuit. If a circuit is layered, its *width* is the maximum number of nodes at any particular layer. We assume that all output nodes appear at the last layer. Every circuit has an equivalent layered circuit obtained by subdividing some edges (using a $g \vee 0$ or $g + 0$ node at a sub-division point g).

Let \mathcal{C} be a complexity class defined in terms of Turing machines. A circuit family $(B_n)_{n \geq 0}$ is said to be \mathcal{C} -uniform, if the direct connection language for B_n can be decided in \mathcal{C} (see [Vollmer 1999](#)).

It is known that languages accepted by Turing machines with a simultaneous space bound $S(n)$ and time bound $T(n)$ can be decided by uniform circuits of depth $O(T(n))$ and width $O(S(n))$ (using the standard tableau construction, see for instance the proof of Theorem 2.10 in [Arora & Barak 2009](#)).

Polynomial-size poly-log depth Boolean circuits of bounded fan-in (every non-leaf node other than a negation node has degree 2) form the class **NC**. Its subclass of log-depth circuits is called **NC**¹, which, with the additional constraint of DLOGTIME uniformity or even log-space uniformity, is known to be contained in **L**. Polynomial-size poly-log width Boolean circuits of bounded fan-in form the class **SC**; **SC**⁰ is the subclass of constant-width circuits and **SC**¹ is the subclass of log-width circuits. It is known that **SC**⁰ equals **NC**¹ ([Barrington 1989](#)) and uniform **SC**¹ equals **L** (via the tableau method). Polynomial-size poly-log depth Boolean circuits with unbounded fan-in form the class **AC**; its subclass of constant

depth circuits is the class AC^0 which is known to be contained in NC^1 . Polynomial-size poly-log depth Boolean circuits with semi-unbounded fan-in (\vee gates have unbounded fan-in, but \wedge gates have fan-in 2) form the class SAC ; in particular, if the depth is $O(\log^i n)$, then the class is SAC^i .

An arithmetic (resp. Boolean) circuit C is said to be *skew* if for every multiplication gate $f = g \times h$ (resp. \wedge gate $f = g \wedge h$), either h or g is a leaf node. C is said to be *weakly skew* if for every $f = g \times h$, either the edge (g, f) or the edge (h, f) is a bridge in the circuit, *i.e.* removing the edge disconnects its end-points in the resulting circuit. Poly-size Boolean skew circuits and weakly skew-circuits are known to characterize NL (Venkateswaran 1992; Toda 1992).

An *algebraic branching program* (BP for short) over a ring \mathbb{K} is a layered directed acyclic graph, where edges are labeled from $\{x_1, \dots, x_n\} \cup \mathbb{K}$. There are two designated nodes, s and t , where s has zero in-degree and t has zero out-degree. The size of a BP is the number of nodes and edges in it, and the width is the maximum number of nodes at any layer. The length of a BP is the length of a longest directed path in it. The polynomial P computed by a BP B is the sum of the weights of all $s - t$ paths in B , where the weight of a path is the product of all edge labels in the path. We will also consider multi-output BPs, where the above is generalized in the obvious way to several nodes t_1, t_2, \dots, t_m at the last level. Note that BPs can be simulated by skew circuits and vice versa with a constant blow up in the width (see, *e.g.*, Toda 1992; Limaye *et al.* 2010).

2.2. Polynomial families. VP denotes the class of families of polynomials $(f_n)_{n \geq 0}$ such that $\forall n \geq 0$

- $f_n \in \mathbb{K}[x_1, \dots, x_{u(n)}]$, where $u \leq \text{poly}(n)$.
- $\deg(f_n) \leq \text{poly}(n)$.
- f_n can be computed by a polynomial-size arithmetic circuit.

VP_e is the sub-class of VP corresponding to poly-size arithmetic formulas (*i.e.* circuits with out-degree at most 1, also called

expressions). VNC^1 is the sub-class of VP containing polynomial families computed by polynomial size, log-depth arithmetic circuits of bounded fan-in. For $i \geq 0$, VSC^i denotes the sub-class of VP , containing polynomial families computed by arithmetic circuits of polynomial size and $\log^i n$ width. It is known that VP_e is the same as VNC^1 (see [Bürgisser 2000](#)). If the circuits computing f_n have quasi-polynomial size $2^{\log^c n}$, we say that $\{f_n\}$ is in the class VQP .

Exponential sums of polynomial families are defined as follows.

DEFINITION 2.1. For a polynomial $g \in \mathbb{K}[X, Y]$, with $X = \{x_1, \dots, x_n\}$, and $Y = \{y_1, \dots, y_m\}$, $E_Y(g)$ denotes the exponential sum of $g(X, Y)$ over all Boolean settings of Y . That is,

$$E_Y(g)(X) = \sum_{e \in \{0,1\}^m} g(X, e).$$

DEFINITION 2.2. Let \mathcal{C} be an algebraic complexity class in Valiant's model. $\Sigma \cdot \mathcal{C}$ is the set of families of polynomials $(f_n)_{n \geq 0}$ such that there exists a polynomial family $(g_m)_{m \geq 0}$ in \mathcal{C} with

$$f_n(X) = E_Y(g_{n+m'})(X) = \sum_{e \in \{0,1\}^{m'}} g_{n+m'}(X, e) \quad \text{where } m' \leq \text{poly}(n).$$

The class VNP is defined to be $\text{VNP} = \Sigma \cdot \text{VP}$.

We denote by VBP and VBWBP the classes of families of polynomials computed by polynomial-size algebraic branching programs of polynomial and constant width, respectively. Without loss of generality, we can treat these classes as polynomial-size skew circuits of polynomial and constant width, respectively ([Malod & Portier 2008](#)).

3. Notion of space for arithmetic computations?

In the case of boolean computations, the notion of “width” of a circuit captures the notion of space in the Turing machine model (under certain uniformity assumptions; see [Pippenger 1979](#)). In the case of arithmetic computations, defining a notion of “space bounded computation” seems to be a hard task.

3.1. Previously studied notions.

Number of registers. One possible measure for space is the number of arithmetic “cells” or registers used in the course of computation (*i.e.* the unit-space model). This measure of space was considered by Valiant (1976) way back in 1976. Later it was again considered by Nisan & Wigderson (1995) in the context of time-space trade-offs for arithmetic straight-line programs. Subsequently, Michaux (1989) showed that with this notion of space, any language that is decided by a machine in the Blum-Shub-Smale (BSS) model of computation (a general model for algebraic computation capturing the idea of computation over reals, Blum *et al.* 1997; see also Bürgisser 2000) can also be computed using $O(1)$ registers. Hence, there is no space-hierarchy theorem under this measure of space. However, Michaux’s result exploits the order relation available in the BSS model of real computation; such a relation is not available in Valiant’s algebraic model. See the appendix for a (very) brief description of the BSS model and Michaux’s result.

Succinct descriptions of polynomials, and weak space.

Another possible measure is the size of a succinct description of the polynomials computed at each cell. de Naurois (2006) introduced a notion of weak space in the Blum-Shub-Smale model and defined the corresponding log space classes LOGSPACE_W and PSPACE_W . This in fact is a way of measuring the complexity of succinctly describing the polynomials computed by or represented at each “real” cell. Though this is a very natural notion of “succinctness” of describing a polynomial, this definition has a few drawbacks:

1. Over \mathbb{R} , it is not known whether NC^1 is contained in LOGSPACE_W . This is in contrast with the situation in the Boolean world.
2. The polynomials representable at every cell have to be “sparse”, *i.e.* the number of monomials with non-zero coefficients should be bounded by some polynomial in the number of variables.

The second condition above makes the notion of weak space very restrictive if we adapt the definition to Valiant’s algebraic

computation model. This is because the corresponding log-space class in this model will be computing only sparse polynomials, but in the non-uniform setting sparse polynomials are known to be contained in a highly restrictive class called skew formulas (see [Section 6](#)), which, as we show in [Corollary 6.2](#), is in fact a proper subclass of constant depth arithmetic circuits (*i.e.* VAC^0).

The Koiran & Perifel model and VPSPACE. Koiran & Perifel ([2009a,b](#)) suggested another notion of polynomial space for Valiant's classes. The main purpose of their definition was to prove a transfer theorem over \mathbb{R} and \mathbb{C} . Under their definition, **Uniform – VPSPACE** is defined as the set of families (f_n) of multivariate polynomials $f_n \in F[x_1, \dots, x_{u(n)}]$ with integer coefficients such that

- $u(n)$ is bounded by a polynomial in n .
- The degree of f_n is bounded by $2^{poly(n)}$.
- Each coefficient of f_n can be represented using $2^{poly(n)}$ bits.
- Every bit of the coefficient function of f_n is computable in PSPACE.

(The non-uniform counterpart can be defined similarly, allowing the PSPACE algorithm for computing coefficients to use polynomially many bits of non-uniform advice.)

Koiran & Perifel ([2009b](#)) observed that the class VPSPACE is equivalent to the class of polynomials computed by uniform arithmetic circuits of polynomial depth and exponential size. Uniform Boolean circuits of polynomial depth and exponential size compute exactly PSPACE, hence the name VPSPACE. Thus, one approach to get reasonable smaller space complexity classes is to generalize this definition. We can consider $VSPACE(s(n))$ to consist of families $(f_n)_{n \geq 1}$ of polynomials satisfying the following:

- $f \in \mathbb{Z}[x_1, \dots, x_{u(n)}]$, where $u(n)$, the number of variables in f_n , is bounded by some polynomial in n .
- The degree of f_n is bounded by $2^{s(n)}$.

- The number of bits required to represent each of the coefficients of f_n is bounded by $2^{s(n)}$, *i.e.* the coefficients of f_n are in the range $[-2^{2^{s(n)}}, 2^{2^{s(n)}}]$.
- Given n in unary, an index $i \in [1, 2^{s(n)}]$, and a monomial M , the i th bit of the coefficient of M in f_n is computable in $\text{DSPACE}(s(n))$.

It is easy to see that with this definition, even the permanent function PERM_n is in $\text{VSPACE}(\log n)$. Thus, $\text{VSPACE}(\log n)$ would be too big a class to be an arithmetic version of log-space. The reason here is that this definition, unlike that of [de Naurois \(2006\)](#), goes to the other extreme of considering only the complexity of coefficient functions and ignores the resource needed to compute and add the monomials with non-zero coefficients. The relationship between the complexity of coefficient functions and the polynomials themselves is explored more thoroughly in [Malod \(2007\)](#).

3.2. Defining VSPACE in terms of circuit width. In this section, we propose width of a (layered) circuit, with additional conditions on the number of variables, the degree and the coefficient size, as a possible measure of space for arithmetic computations. Note that treating width as space is essentially the register model, with manipulations of register contents allowed using only the ring operations.

DEFINITION 3.1. *For any $S : \mathbb{N} \rightarrow \mathbb{N}$ such that $S(n) \geq n, \forall n$, $\text{VWIDTH}(S)$ is the class of polynomial families $(f_n)_{n \geq 0} \in \mathbb{Z}[X]$ (with integer coefficients) with the following properties:*

- $f_n \in \mathbb{Z}[x_1, \dots, x_{u(n)}]$, where $u(n)$, the number of variables in f_n is at most $\text{poly}(n)$.
- $\deg(f) \leq \max\{2^{S(n)}, \text{poly}(n)\}$.
- The coefficients of f_n are representable using $\max\{2^{S(n)}, \text{poly}(n)\}$ many bits.
- There is an arithmetic circuit of width $S(n)$ and size $\max\{2^{S(n)}, \text{poly}(n)\}$ computing f_n .

Further, if the arithmetic circuits in the last condition are $DSPACE(S)$ -uniform, we call the class $Uniform-VWIDTH(S)$.

We show now that at polynomial space, this definition is equivalent to that of [Koiran & Perifel \(2009b\)](#).

THEOREM 3.2. *The class $Uniform-VPSPACE$ as defined in [Koiran & Perifel \(2009b\)](#) coincides with $Uniform-VWIDTH(poly)$.*

We use the following easy fact:

FACT 3.3. *A degree d polynomial over t variables has at most $\binom{d+t}{t}$ monomials.*

Now, [Theorem 3.2](#) follows from the two lemmas below.

LEMMA 3.4. $Uniform-VPSPACE \subseteq Uniform-VWIDTH(poly)$.

PROOF. Let $(f_n)_{n \geq 0}$ be a family of polynomials in $VPSPACE$. Then by definition, the bits of the coefficients of f_n can be computed in $PSPACE$ and hence by exponential-size Boolean circuits of polynomial width. The (exponentially many) bits can be put together with appropriate weights to obtain an arithmetic circuit computing the coefficient itself. The exponential-degree monomials can each be computed by an exponential-size constant-width circuit. Thus, we can use the naive method of computing f_n : expand f_n into individual monomials, compute each coefficient and each monomial, and add them up sequentially. By [Fact 3.3](#), there are only exponentially many distinct monomials. Thus, we get a polynomial-width exponential-size circuit computing f_n . \square

The converse direction is a little more tedious, but essentially follows from the Lagrange interpolation formula for multivariate polynomials.

LEMMA 3.5. $Uniform-VWIDTH(poly) \subseteq Uniform-VPSPACE$.

PROOF. Let $(f_n)_{n \geq 0}$ be a family of polynomials in $VWIDTH(poly(n))$. Let $N = u(n)$ be the number of variables in f_n , and let $q(n)$ be a polynomial such that $2^{q(n)}$ is an upper bound on both $d = \deg(f_n)$ and on the number of bits required to represent each

coefficient. Let $w(n) = \text{poly}(n)$ and $s(n) \in 2^{O(n^c)}$ respectively be the width and size of a witnessing arithmetic circuit C that computes f_n .

To show that $f_n \in \text{VPSPACE}$, we need to give a PSPACE algorithm which, given 1^n and $\langle i_1, \dots, i_N \rangle$ as input, computes the coefficient of the monomial $\prod_{k=1}^N x_k^{i_k}$.

We use the following notation: $S = \{0, 1, \dots, d\}$, $T = S^N$, $\tilde{x} = \langle x_1, \dots, x_N \rangle$, and for $\tilde{i} = \langle i_1, \dots, i_N \rangle \in T$, the monomial $m(\tilde{i}) = \prod_{k=1}^N x_k^{i_k}$ is denoted $\tilde{x}^{\tilde{i}}$. We drop the subscript n in f_n for convenience.

Using Lagrangian interpolation for multivariate polynomials, we have

$$f(\tilde{x}) = \sum_{\tilde{i} \in T} f(\tilde{i}) \text{Equal}(\tilde{x}, \tilde{i}) = \sum_{\tilde{i} \in T} f(\tilde{i}) \prod_{k=1}^N \text{Equal}(x_k, i_k),$$

$$\text{where } \text{Equal}(x, i) = \prod_{a \in S \setminus \{i\}} \left(\frac{x - a}{i - a} \right) = \frac{\prod_{a \in S \setminus \{i\}} (x - a)}{i!(d - i)!(-1)^{d-i}}.$$

Thus, for any $\tilde{t} \in T$, the coefficient of the monomial $m(\tilde{t})$ is given by

$$\text{coeff}(m(\tilde{t})) = \sum_{\tilde{i} \in T} f(\tilde{i}) \prod_{k=1}^N \frac{\text{coeff of } x_k^{t_k} \text{ in } \prod_{a \in S \setminus \{i_k\}} (x_k - a)}{i_k!(d - i_k)!(-1)^{d-i_k}}.$$

But we have a nice form for the inner numerator:

$$\begin{aligned} &\text{coeff of } x_k^{t_k} \text{ in } \prod_{a \in S \setminus \{i_k\}} (x_k - a) \text{ is} \\ &(-1)^{d-t_k} S_{d, d-t_k}(0, 1, \dots, i_k - 1, i_k + 1, \dots, d), \end{aligned}$$

where $S_{d,j}$ denotes the elementary symmetric polynomial of degree j in d variables.

To compute the desired coefficient in PSPACE, we use the Chinese Remaindering technique; See [Chiu *et al.* \(2001\)](#) for more details. Since symmetric polynomials are easy to compute (*e.g.* [Shpilka & Wigderson 2002](#) or Theorem 2.5.4 in [Tzamaret 2008](#)), and since $f(\tilde{i})$ is computable by a polynomial-width arithmetic

circuit by assumption, a PSPACE algorithm can compute the coefficient modulo a prime p , for any prime p that has an $O(d)$ bit representation. (The algorithm will require $O(w(n) \log p + \log s(n))$ space to evaluate $f(\tilde{i}) \bmod p$). Reconstructing the coefficient from its residues modulo all such primes can also be performed in PSPACE (see Chiu *et al.* 2001) \square

REMARK 3.6. *It is straightforward to see that Lemma 3.4 holds without the uniformity condition, i.e. $\text{VSPACE} \subseteq \text{VWIDTH}(\text{poly})$. However, it is not clear if the same is true for Lemma 3.5. If a polynomial family is computed by non-uniform arithmetic circuits of polynomial width and exponential size, then we do not know how to compute coefficient functions in PSPACE using only polynomially many bits of advice. (The advice is not long enough to encode the circuit.)*

3.3. $\text{VWIDTH}(S)$ for sub-linear S . Motivated by the equivalence in Theorem 3.2, we now consider using Definition 3.1 for sub-linear functions, specifically, for poly-logarithmic space. We immediately run into a problem; for $i \geq 2$, $\text{VWIDTH}(\log^i n)$ and VSC^i , though close, are different for the following reasons:

- Polynomials in $\text{VWIDTH}(\log^i n)$ can have degree $O(2^{\log^i n})$, whereas degree of polynomials in VSC^i is bounded by $\text{poly}(n)$. Since we are concerned with polynomials in VNP , we can simply change the degree constraint in VWIDTH to a polynomial upper bound. But then Theorem 3.2 will not go through. With the current definition, however, this is not a problem for VSC^0 and VSC^1 .
- The coefficients of polynomials in $\text{VWIDTH}(\log^i n)$ are integers and their size is bounded by $O(2^{\log^i n})$, whereas polynomials in VSC^i can have arbitrary coefficients from the underlying ring. This is not a problem if we consider VSC circuits where leaf labels are from the set $X \cup \{0, 1\}$ or $X \cup \mathbb{Z}$. However, the main reason for the integer coefficients with polynomial bit size in Definition 3.1 and in Koiran & Perifel (2009b) is to allow a simulation by uniform PSPACE machines. Since we are not directly concerned with such a simulation here, we can alternatively

change the coefficient constraint in [Definition 3.1](#) to allow arbitrary ring elements. In this case, the constraint on coefficient size should also be dropped.

We thus consider the following definition for sub-linear functions:

DEFINITION 3.7. *For a sub-linear function $S : \mathbb{N} \longrightarrow \mathbb{N}$, $\text{VWIDTH}(S)$ is the class of polynomial families $(f_n)_{n \geq 0}$ with the following properties:*

- *The number of variables $u(n)$ in f_n is bounded by $\text{poly}(n)$.*
- *$\deg(f_n) \leq \max\{2^{O(S(n))}, \text{poly}(n)\}$.*
- *There is a family of arithmetic circuits of size $\max\{2^{O(S(n))}, \text{poly}(n)\}$ and width $O(S(n))$ computing (f_n) .*

Then we have $\text{VWIDTH}(\log^0 n) = \text{VSC}^0$ and $\text{VWIDTH}(\log n) = \text{VSC}^1$.

We now define the following algebraic complexity classes:

DEFINITION 3.8. $\text{VSPACE}(S(n)) \triangleq \text{VWIDTH}(S(n))$.

$\text{Uniform-VSPACE}(S(n)) \triangleq \text{Uniform-VWIDTH}(S(n))$.

We denote the log-space class by VL ; thus, $\text{VL} = \text{VWIDTH}(\log n) = \text{VSC}^1$.

The following containments and equalities follow directly from known results (see e.g., [Caussinus et al. 1998](#); [Limaye et al. 2010](#); [Flarup & Lyaudet 2010](#)) about width-constrained arithmetic circuits.

LEMMA 3.9. $\text{VBWBP} = \text{VNC}^1 = \text{VP}_e \subseteq \text{VSPACE}(O(1)) = \text{VSC}^0 \subseteq \text{VL} = \text{VSC}^1 \subseteq \text{VP}$.

Thus, VL according to this definition is in VP and avoids the trivially “too-powerful” trap; also, it contains VNC^1 and thus avoids the “too weak” trap.

The following closure property is easy to see.

LEMMA 3.10. *For every $S(n) \geq \log n$, the classes $\text{VSPACE}(S(n))$ are closed under polynomially bounded summations and products.*

4. Read-once certificates

In general, non-deterministic complexity classes can be defined via existential quantifiers. *e.g.*, $\text{NP} = \exists \cdot \text{P}$. In the algebraic setting, we know that the class VNP (algebraic counterpart of NP) is defined as an “exponential” sum of values of a polynomial-size arithmetic circuit, *i.e.* $\text{VNP} = \Sigma \cdot \text{P}$. It is also known that $\text{VNP} = \Sigma \cdot \text{VP}_e = \Sigma \cdot \text{VNC}^1$ (see [Bürgisser 2000](#)).

If we consider smaller classes, NL is the natural non-deterministic version of L . However, to capture it via existential quantifiers, we need to restrict the use of the certificate, since otherwise $\exists \cdot \text{L} = \text{NP}$. It is known that with the notion of “read-once” certificates (see, *e.g.*, [Arora & Barak 2009](#), Chapter 4) one can express NL as an existential quantification over L . Analogously, we propose a notion of “read-once” certificates in the context of arithmetic circuits so that we can get meaningful classes by taking exponential sums over classes that are below VP .

DEFINITION 4.1. *Let C be a layered arithmetic circuit with ℓ layers. Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ be the input variables of C . C is said to be “read-once certified” in Y if the layers of C can be partitioned into m blocks, such that each block reads exactly one variable from Y . That is, there is a fixed permutation $\pi \in S_m$, and indices $0 = i_1 \leq \dots \leq i_m \leq i_{m+1} = \ell$, such that for each $1 \leq j \leq m$, all occurrences of the variable $y_{\pi(j)}$ are at gates appearing in layers $i_j + 1$ to i_{j+1} .*

Without loss of generality, we henceforth assume that π is the identity permutation.

Now, analogous to [Definition 2.1](#), [Definition 2.2](#), we define the exponential sum over read-once certified circuits.

DEFINITION 4.2. *Let \mathcal{C} be any arithmetic circuit complexity class. A polynomial family $(f_n)_{n \geq 0}$ is said to be in the class $\Sigma^R \cdot \mathcal{C}$, if there is a family $(g_{m(n)})_{n \geq 0}$ such that $m(n) = n + m'(n)$, $m'(n) \leq \text{poly}(n)$, $f_n(X) = E_Y(g_{m(n)})$ and $g_{m(n)}$ can be computed by a circuit of type \mathcal{C} that is read-once certified in Y .*

We also use the term “read-once exponential sum over \mathcal{C} ” to denote $\Sigma^R \cdot \mathcal{C}$.

For circuits of width polynomial or more, the restriction to read-once certification is immaterial: the circuit can read a variable once and carry its value forward to any desired layer via internal gates. This is equivalent to saying that for a P machine, read-once input is the same as two-way-readable input. Thus,

PROPOSITION 4.3. $\Sigma^R \cdot \text{VP} = \Sigma \cdot \text{VP} = \text{VNP}$.

Having seen that the read-once certificate definition is general enough for the case of large width circuits, we turn our focus on circuits of smaller width. Once the width of the circuit is substantially smaller than the number of bits in the certificate, the read-once property becomes a real restriction. If this restriction correctly captures non-determinism, we would expect that in analogy to $\text{BP} = \text{NL} = \exists^R \cdot \text{L}$, we should be able to show that VBP equals $\Sigma^R \cdot \text{VL}$. In a partial answer, we show in the following theorem one direction: read-once exponential sums over VL are indeed powerful enough to contain VBP .

THEOREM 4.4. $\text{VBP} \subseteq \Sigma^R \cdot \text{VL}$.

In order to prove the above theorem, we consider a problem that is complete for VBP . We need the following definition:

DEFINITION 4.5. A polynomial $f \in \mathbb{K}[X_1, \dots, X_n]$ is called a *projection* of g (denoted $f \leq g$) if $f(X_1, \dots, X_n)$ is identically equal to $g(a_1, \dots, a_m)$, where each a_i is a ring element or a variable, that is, each $a_i \in \mathbb{K} \cup \{X_1, \dots, X_n\}$.

Let $f = (f_n)_{n \geq 0}$ and $g = (g_m)_{m \geq 0}$ be two polynomial families. f is said to be *projection reducible* to g if

$$\exists n_0 : \forall n \geq n_0, f_n \leq g_{m(n)} \quad \text{where } m(n) \leq \text{poly}(n).$$

Let $(G_n) = (V_n, E_n)$ be a family of directed acyclic graphs defined as follows:

1. G_n is a layered graph with $n + 1$ layers. For $1 \leq i \leq n + 1$, the i th layer contains n vertices, denoted v_1^i, \dots, v_n^i . (So totally there are $n(n + 1)$ vertices.)

2. For $1 \leq i, j, k \leq n$, there is a directed edge (v_j^i, v_k^{i+1}) , labeled by the variable $x_{i,j,k}$.

For a directed path P in G_n , let $w(P)$ denote the monomial obtained by taking the product of the labels of the edges in P . Let $s = v_1^1$ and $t = v_1^{n+1}$. We define the polynomial family $PATH$ as follows:

$$PATH_n = \sum_{P: P \text{ is a directed } s-t \text{ path in } G_n} w(P).$$

Note that $PATH_n$ is a polynomial in n^3 variables. It is easy to see the following:

PROPOSITION 4.6. (*folklore*) $PATH$ is complete for VBP under projections.

We prove [Theorem 4.4](#) by showing that $PATH \in \Sigma^R \cdot \text{VL}$.

PROOF (of [Theorem 4.4](#)). Here onwards we drop the index n from G_n .

We define function $h_G(Z) : \{0, 1\}^{n^3} \rightarrow \{0, 1\}$ as follows. We can think of the variables in $Z = \{Z_{1,1,1}, \dots, Z_{n,n,n}\}$ as picking a subset of the edges of G . The function $h_G(Z)$ evaluates to 1 if and only if this subset is exactly a directed $s-t$ path in G . Note that $s-t$ paths P in G are in one-to-one correspondence with assignments to Z such that $h_G(Z) = 1$. Hence,

$$\begin{aligned} PATH_n &= \sum_{\substack{P: P \text{ is a} \\ \text{directed } s-t \text{ path}}} w(P) \\ &= \sum_{z \in \{0,1\}^{n^3}} h_G(z) [\text{weight of edges picked by } Z] \\ (4.7) \quad &= \sum_{z \in \{0,1\}^{n^3}} h_G(z) \prod_{i,j,k} [x_{i,j,k} z_{i,j,k} + (1 - z_{i,j,k})]. \end{aligned}$$

Provided that the bits of z are given in the correct order (proceeding layer by layer), $h_G(z)$ can be computed in deterministic log-space, with z given on a read-once input tape, as follows.

1. Input $1^n, z \in \{0, 1\}^{n^3}$.
2. Initialize $current := 1$ (because s is the first node in its layer).
3. For $i = 1$ to n repeat steps 4 and 5.
4. Find a j such that $z_{i,current,j} = 1$.
If there is no such j or more than one such j then REJECT.
5. Set $current := j$.
6. If $current = 1$ then ACCEPT (because s is the first node in its layer) otherwise REJECT.

The algorithm A above is deterministic and uses a total of $O(\log n)$ bits of work-space. For a fixed n , let C be the $O(\log n)$ width boolean circuit corresponding to A . (Without loss of generality, assume that all negation gates in C are at the leaves. If this is not already the case, transforming C to ensure this only doubles the width, and does not destroy the read-once property.) Let D be the natural arithmetization of C . Since Z is on a read-once input tape, C , and hence D , are read-once certified in the variables from Z . We can attach, parallel to D , constant-width circuitry that collects factors of the product $\prod_{i,j} (x_{i,j,k} z_{i,j,k} + (1 - z_{i,j,k}))$ as and when the $z_{i,j,k}$ variables are read, and finally multiplies this with the computed value $h_G(Z)$. The resulting circuit remains $O(\log n)$ -width and remains read-once certified on Z . From ((4.7)), it follows that the read-once exponential sum of D over bit assignments for Z computes the polynomial $PATH_n$. \square

Not only are we unable to show the converse, we are also unable to show a reasonable upper bound on $\Sigma^R \cdot \text{VL}$. It is not even clear if $\Sigma^R \cdot \text{VL}$ is contained in VP . One possible interpretation is that the Σ^R operator is too powerful and can lift up small classes unreasonably. We show that this is not the case in general; in particular, it does not lift up VBP and VBWBP .

THEOREM 4.8. (i) $\Sigma^R \cdot \text{VBP} = \text{VBP}$.

(ii) $\Sigma^R \cdot \text{VBWBP} = \text{VBWBP}$.

To prove this theorem, we first state and prove [Lemma 4.9](#).

LEMMA 4.9. *Let C be a layered skew arithmetic circuit on variables $X \cup Y$ that is read-once certified in Y . Let $w = \text{width}(C)$, $s = \text{size}(C)$ and $m = |Y|$. Let f_1, \dots, f_w denote the output gates (also the polynomials computed by them) of C . There exists a weakly skew circuit C' , of size $56mw^3s$ and width $7w$, that computes all the exponential sums $E_Y(f_1), \dots, E_Y(f_w)$.*

PROOF. We proceed by induction on $m = |Y|$. In the base case when $m = 1$, $E_Y(f_j)(X) = f_j(X, 0) + f_j(X, 1)$. Putting two copies of C next to each other, one with $y = 0$ and the other with $y = 1$ hardwired, and adding the corresponding outputs, we get a circuit C' which computes the required function. Clearly $\text{width}(C') \leq 2w$ and $\text{size}(C') \leq 2s + 1$.

Assume now that the lemma is true for all skew circuits with $m' = |Y| < m$. Let C be a given circuit where $|Y| = m$. Let Y' denote $Y \setminus \{y_m\} = \{y_1, \dots, y_{m-1}\}$. As per Definition 4.1, the layers of C can be partitioned into m blocks, with the k th block reading only y_k from Y . Let $0 = i_1 \leq i_2 \leq \dots \leq i_m \leq i_{m+1} = \ell$ be the layer indices such that y_k is read between layers $i_k + 1$ and i_{k+1} . Let f_1, \dots, f_w be the output gates of C .

We slice C into two parts: the bottom $m - 1$ blocks of the partition together form the circuit D , and the top block forms the circuit C_m . Let g_1, \dots, g_w be the output gates of D . These are also the inputs to C_m ; we symbolically relabel the non-leaf inputs at level 0 of C_m as z_1, \dots, z_w and the outputs of C_m as h_1, \dots, h_w . Clearly, C_m and D are both skew circuits of width w . Further, each h_j depends on X, y_m and $Z = \{z_1, \dots, z_w\}$; that is, $h_1, \dots, h_w \in R[Z]$ where $R = \mathbb{K}[X, y_m]$. Similarly, each g_j depends on X and Y' ; $g_1, \dots, g_w \in \mathbb{K}[X, Y']$. The values computed by C can be expressed as $f_j(X, Y) = h_j(X, y_m, g_1(X, Y'), \dots, g_w(X, Y'))$.

Since C and C_m are skew circuits, and since the variables z_j represent non-leaf gates of C , C_m is linear in these variables. Hence each h_j can be written as $h_j(X, y_m, Z) = c_j + \sum_{k=1}^w c_{j,k} z_k$, where the coefficients $c_j, c_{j,k} \in \mathbb{K}[X, y_m]$. Combining this with the expression for f_j , we have

$$\begin{aligned}
f_j(X, Y) &= h_j(X, y_m, g_1(X, Y'), \dots, g_w(X, Y')) \\
&= c_j(X, y_m) + \sum_{k=1}^w c_{j,k}(X, y_m) g_k(X, Y').
\end{aligned}$$

Hence

$$\begin{aligned}
\sum_{e \in \{0,1\}^m} f_j(X, e) &= \sum_{e=(e', e_m) \in \{0,1\}^m} \left[c_j(X, e_m) + \sum_{k=1}^w c_{j,k}(X, e_m) g_k(X, e') \right] \\
&= 2^{m-1} \sum_{e_m=0}^1 c_j(X, e_m) + \sum_{k=1}^w \sum_{e' \in \{0,1\}^{m-1}} c_{j,k}(X, e_m) g_k(X, e') \\
&= 2^{m-1} \sum_{e_m=0}^1 c_j(X, e_m) + \\
&\quad \sum_{k=1}^w \left(\sum_{e_m \in \{0,1\}} c_{j,k}(X, e_m) \right) \left(\sum_{e' \in \{0,1\}^{m-1}} g_k(X, e') \right).
\end{aligned}$$

$$\text{Thus } E_Y(f_j)(X) = 2^{m-1} E_{y_m}(c_j)(X) + \sum_{k=1}^w E_{y_m}(c_{j,k})(X) E_{Y'}(g_k)(X).$$

By induction, we know that there is a weakly skew circuit D' of width $7w$ and size $56(m-1)w^3s$ computing $E_{Y'}(g_k)(X)$ for all k simultaneously.

To compute $E_{y_m}(c_j)(X)$, note that a copy of C_m with all leaves labeled from Z replaced by 0 computes exactly $c_j(X, y_m)$. So the sum $E_{y_m}(c_j)(X)$ can be computed as in the base case, in width $2w$. Since we need only one sum as opposed to w simultaneous sums, the size bound is $2\text{size}(C_m) + 1$. Multiplying this by 2^{m-1} adds nothing to width and 1 to size, so the overall width of this skew circuit is $2w$ and the size is at most $2s + 2 \leq 3s$.

To compute $E_{y_m}(c_{j,k})(X)$, we modify C_m as follows: replace leaves labeled z_k by the constant 1, replace leaves labeled $z_{k'}$ for $k' \neq k$ by 0, leave the rest of the circuit unchanged, and let h_j be the output gate. This circuit computes $c_j(X, y_m) + c_{j,k}(X, y_m)$. Subtracting $c_j(X, y_m)$ re-computed as above from this gives $c_{j,k}(X, y_m)$, computed in width $2w$ and size $2s + 2$. (The subtraction needs two gates: a $\times -1$ and a $+$.) Now, again the sum $E_{y_m}(c_{j,k})(X)$ can be

computed as in the base case; we use two copies of the difference circuit with $y_m = 0$ and $y_m = 1$ hardwired, and add their outputs. This gives a skew circuit of width $4w$ and size $2(2s + 2) + 1 \leq 5s$.

We now use these skew circuits, and the weakly skew circuit D' available by induction, to construct the desired circuit. Since we use fresh copies of C for each of the circuits for c_j and $c_{j,k}$, the resulting circuit is weakly skew.

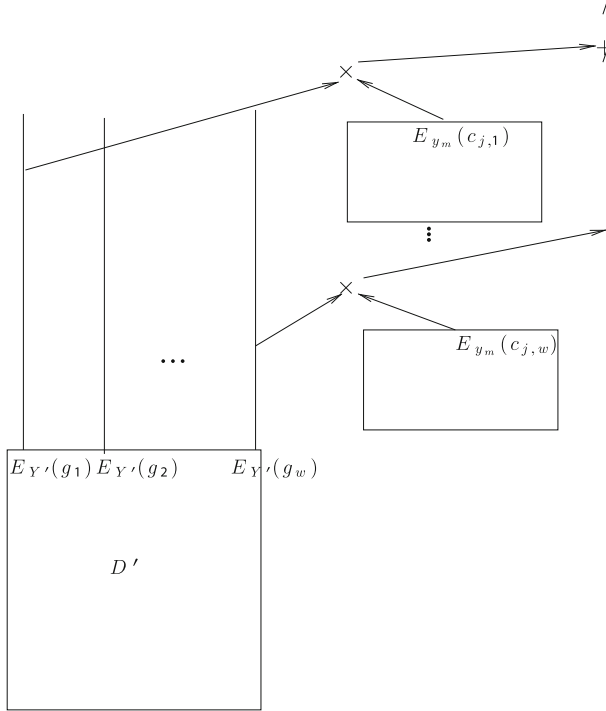
Putting together these circuits naively may increase width too much. So we position D' at the bottom and carry w wires upwards from it corresponding to its w outputs. Alongside these wires, we position circuitry to accumulate the terms for each f_j and to carry forward already-computed f_k 's. The width in this part is w for the wires carrying the outputs of D' , w for wires carrying the values $E_Y(f_j)$, $4w$ for computing the terms in the sum above (they are computed sequentially so the width does not add up), and 1 for carrying the partial sum in this process, overall at most $6w + 1 \leq 7w$. Thus, the resulting circuit, which is the desired circuit D , has width at most $\max\{\text{width}(D'), 7w\} = 7w$. Figure 4.1 shows how this is done for one of the sums $E_Y(f_j)$.

To bound the size of the circuit, we bound its depth in the part above D' by d ; then size is at most $\text{size}(D') + \text{width} \times d$. The circuit has w modules, one each to compute each of the $E_Y(f_j)$ s. The depth of each module can be bounded by the depth needed to compute $E_{y_m}(c_j)$, plus w times the depth to compute any one $E_{y_m}(c_{j,k})$, that is, at most $3s + w \times 5s$. So $d \leq w(3s + 5sw) \leq 8w^2s$. Hence

$$\begin{aligned} \text{size}(D) &\leq \text{size}(D') + (\text{width above } D') \times (\text{depth above } D') \\ &\leq 56(m - 1)w^3s + 7w(8w^2s) \\ &\leq 56mw^3s. \end{aligned} \quad \square$$

Now we prove Theorem 4.8:

PROOF. We start with the VBP or VBWBP for which we want to compute the read-once exponential sum. We treat it as a read-once skew circuit and apply Lemma 4.9 to get a weakly skew circuit for the sum.

Figure 4.1: Computation of $E_Y(f_j)$ in a width-efficient manner

1. From [Jansen \(2008\)](#), [Kaltofen & Koiran \(2008\)](#), we know that weakly skew circuits can be transformed into skew circuits and hence branching programs with a constant blowup in the size. This gives the desired equivalence.
2. From [Jansen & Rao \(2009\)](#), we know that when the width of the weakly skew circuit is a constant, it can be transformed into a skew circuit whose width is again a constant, and the size blowup is polynomial. This gives the desired equivalence in this case. \square

5. Read-once exponential sums of multiplicatively disjoint circuits

In this section, we explore how far the result of [Theorem 4.8](#) can be pushed to larger classes within VP. In effect, we ask whether

the technique of [Lemma 4.9](#) is applicable to larger classes of circuits. Such a question is relevant because we do not have any upper bound (better than VNP) even for $\Sigma^R \cdot \text{VSC}^0$ and $\Sigma^R \cdot \text{VL}$.

The generalization we consider is multiplicative disjointness. An arithmetic circuit C is said to be *multiplicatively disjoint* (**md**-) if every multiplication gate operates on sub-circuits which are not connected to each other. This is a relaxation of the weakly skew condition, since the sub-circuits are allowed to have multiple connections to gates above the concerned multiplication gate. This restriction was first considered in [Malod & Portier \(2008\)](#), where it is shown that multiplicatively disjoint polynomial-size circuits characterize VP .

Examining the proof of [Lemma 4.9](#), we see that the main barrier in extending it to the larger class of **md**-circuits is that when we slice C into D and C_m , C_m may not be linear in the “slice variables” Z . However, for **md**-circuits, C_m is multilinear in Z . As far as computing the coefficients $c_{j,\alpha}$ goes, where α describes a multilinear monomial, this is not a problem; it can be shown that for such circuits the coefficient function can be computed efficiently. There is a cost to pay in size because the number of multilinear monomials is much larger. To handle this, we modify the inductive step, slicing C not at the last block but at a level that halves the number of Y variables read above and below it. This works out fine for constant width, but results in quasi-polynomial blow-up in size for larger widths.

We show the following:

THEOREM 5.1. (i) $\Sigma^R \cdot \text{md-VSC}^0 \subseteq \text{VP}$.

(ii) $\Sigma^R \cdot \text{md-VSC} \subseteq \text{VQP}$.

The high-level strategy for proving [Theorem 5.1](#) is as follows.

1. Break the circuit by a horizontal cut into two parts A and B , so that each part contains approximately $m/2$ variables from Y *i.e.* $Y_A, Y_B \leq \lceil m/2 \rceil$ and $Y_A \cup Y_B = Y, Y_A \cap Y_B = \emptyset$. Let A be the upper part.

2. Now express the polynomials in A as sums of monomials where the variables stand for the output gates of B and the coefficients come from $\mathbb{K}[X, Y_A]$.
3. Inductively compute the E_Y 's for the coefficients of A and the monomials in terms of the output gates of B .
4. Apply (5.3) (from Observation 5.2) to obtain the required $E_Y(f_j)$ s.

This strategy is spelt out in detail in Lemma 5.4. Theorem 5.1 follows directly from it.

We need the following observation (which is already used implicitly in the proof of Lemma 4.9):

OBSERVATION 5.2. (i) If $f = g + h$, then $E_Y(f) = E_Y(g) + E_Y(h)$.

(ii) If $f = g \times h$ and if the variables of Y can be partitioned into Y_g and Y_h such that g depends only on $X \cup Y_g$ and h depends only on $X \cup Y_h$, then

$$(5.3) \quad E_Y(f) = E_{Y_g}(g) \times E_{Y_h}(h).$$

LEMMA 5.4. Let C be a layered multiplicatively disjoint circuit of width w and size s on variables $X \cup Y$, and let $m = |Y|$. Let ℓ be the number of layers in C . Suppose C is read-once certified in Y . Let f_1, \dots, f_w be the output gates of C . Then, there is an arithmetic circuit C' of size $T(w, m, s) \leq 3sm^{cw}$ which computes $E_Y(f_1), \dots, E_Y(f_w)$, where c is an absolute constant. (Any $c \geq \frac{5}{3-\log 7}$ suffices.)

PROOF. The proof is by induction on $m = |Y|$.

In the base case when $m = 1$, C' has two copies of C with $y_1 = 0$ and $y_1 = 1$ hard-wired, and one $+$ gate for each output. The resulting size is $2s + w \leq 3s$, since $w \leq s$.

Assume the induction hypothesis: For any arithmetic circuit D on variables $X \cup Y'$ of size s' and width w' , with $|Y'| = m' < m$,

there is an arithmetic circuit D' of size $T(w', m', s')$ computing $E_Y(f'_1), \dots, E_Y(f'_{w'})$, where $f'_1, \dots, f'_{w'}$ are the output gates of D .

Let $0 = i_1 \leq i_2 \leq \dots \leq i_m \leq i_{m+1} = \ell$ be the level indices of C as guaranteed by [Definition 4.1](#). Consider level $\ell' = i_{\lceil m/2 \rceil + 1}$. Let g_1, \dots, g_w be the gates at level ℓ' .

We slice C at level ℓ' ; the circuit above this level is A and the circuit below it is called B . In particular, A is obtained from C by re-labeling the gates g_1, \dots, g_w with new variables z_1, \dots, z_w and removing all gates below level ℓ' . Let h_1, \dots, h_w denote the output gates of A . (Note that these are just relabellings of f_1, \dots, f_w .) Similarly, B is obtained from C by removing all nodes above layer ℓ' and making g_1, \dots, g_w the output gates. Let s_A and s_B respectively denote their sizes. Let $Y_A \subseteq Y$ (resp. Y_B) be the set of variables from Y that appear in A (resp. B). The circuits A and B have the following properties:

1. A and B are multiplicatively disjoint and are of width w .
2. A is *syntactically multilinear* in the variables $Z = \{z_1, \dots, z_w\}$: at every \times gate $f = g \times h$, each variable in Z has a path to g or to h or to neither, but not to both.
3. $Y_A \cap Y_B = \emptyset, Y_A \cup Y_B = Y, |Y_A| = \lfloor m/2 \rfloor$ and $|Y_B| = \lceil m/2 \rceil$.
4. For $1 \leq j \leq w, g_j \in \mathbb{K}[X, Y_B]$ and $h_j \in R[Z]$, where $R = \mathbb{K}[X, Y_A]$.
5. Let $v = v_1 \times v_2$ be a multiplication gate in A . If there is a path from z_i to v_1 and there is a path from z_j ($i \neq j$) to v_2 , then the sub-circuits of C (and hence of B) rooted at g_i and g_j are disjoint.

Since A is syntactically multilinear in Z and C is md- , the monomials in $h_j \in R[Z]$ can be described by subsets of Z , where z_i and z_k can belong to a subset corresponding to a monomial only if the sub-circuits rooted at g_i and g_k are disjoint. Let S denote the subsets that can possibly correspond to monomials:

$$S = \left\{ S' \subseteq Z \mid \forall z_i, z_k \in S' \text{ with } i \neq k, \text{ the sub-circuits rooted at } g_i \text{ and } g_k \text{ are disjoint} \right\}.$$

Generally, we treat S as a set of characteristic vectors instead of actual subsets; the usage will be understood from the context.

We can express the polynomials computed by A and C as follows:

$$f_j = h_j(g_1, \dots, g_w); \quad h_j = \sum_{\alpha \in S} c_{j,\alpha} Z^\alpha,$$

$$\text{where } Z^\alpha = \prod_{i=1}^w z_i^{\alpha_i} \text{ and } c_{j,\alpha} \in \mathbb{K}[X, Y_A].$$

$$\text{Hence } f_j(X, Y) = \sum_{\alpha \in S} c_{j,\alpha}(X, Y_A) g^\alpha(X, Y_B)$$

$$\text{where } g^\alpha(X, Y_B) = \prod_i g_i^{\alpha_i}(X, Y_B).$$

Using [Observation 5.2](#), we have

$$(5.5) \quad E_Y(f_j) = \sum_{\alpha \in S} E_Y(c_{j,\alpha} g^\alpha) = \sum_{\alpha \in S} E_{Y_A}(c_{j,\alpha}) E_{Y_B}(g^\alpha).$$

We need the following claim:

CLAIM 5.6. *For $1 \leq j \leq w$, and for $\alpha \in S$, the polynomial $c_{j,\alpha}(X, Y_A)$ can be computed by a multiplicatively disjoint circuit $[c_{j,\alpha}]$ of size $w \cdot s_A$ and width w . Moreover, $[c_{j,\alpha}]$ is read-once certified in Y_A .*

PROOF. We build an arithmetic circuit $[c_{j,\alpha}]$ for $c_{j,\alpha}(X, Y_A)$ by induction on the structure of the circuit rooted at h_j . Let $\alpha = \alpha_1 \alpha_2 \dots \alpha_w$, where $\alpha_i \in \{0, 1\}$.

1. Base case: The sub-circuit rooted at h_j is a variable z_i or an element $a \in \mathbb{K} \cup X \cup Y_A$. Then $[c_{j,\alpha}]$ is set accordingly as follows:

$$\text{If } h_j = z_i, \quad \text{then} \quad [c_{j,\alpha}] = \begin{cases} 1 & \text{if } \alpha_i = 1, \text{ and } \alpha_k = 0 \ \forall i \neq k; \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{If } h_j = a \in (\mathbb{K} \cup X \cup Y_A), \quad \text{then} \quad [c_{j,\alpha}] = \begin{cases} a & \text{if } \alpha_i = 0 \ \forall i; \\ 0 & \text{otherwise.} \end{cases}$$

2. Induction step:

Case 1: $h_j = h_j^1 + h_j^2$. Then $[c_{j,\alpha}] = [h_{j,\alpha}^1] + [h_{j,\alpha}^2]$.

Case 2: $h_j = h_j^1 \times h_j^2$. Then $[c_{j,\alpha}] = [h_{j,\alpha'}^1] \times [h_{j,\alpha''}^2]$, where α' (respectively α'') is α restricted to the Z -variables that appear at the sub-circuit rooted at h_1 (respectively h_2). We set $[c_{j,\alpha}]$ to 0 if α' and α'' do not form a partition of α . Note that $[h_{j,\alpha}^1], [h_{j,\alpha}^2], [h_{j,\alpha'}^1]$ and $[h_{j,\alpha''}^2]$ are the corresponding coefficients available from the inductive hypothesis.

The size of $[c_{j,\alpha}]$ thus obtained can blow-up by a factor of at most w , and the width remains unchanged. The circuit $[c_{j,\alpha}]$ is a projection of the sub-circuit rooted at h_j , obtained by replacing variables z_i by constants 0 or 1. Since the sub-circuit rooted at h_j is read-once certified in Y_A , so is $[c_{j,\alpha}]$. \square

If $\alpha \in S$, then g^α can be computed by an md-circuit of width w and size $s_B + w$. Let $[g^\alpha]$ denote this circuit. (It is a multiplicative circuit sitting on top of the relevant output gates of B .)

By the induction hypothesis, the polynomials $E_{Y_A}(c_{j,\alpha})$ for $1 \leq j \leq w$ and $\alpha \in S$ can be computed by arithmetic circuits of size $T(w, \lfloor m/2 \rfloor, ws_A)$. Also, by induction, the polynomials $E_{Y_B}(g^\alpha)$ can be computed by arithmetic circuits of size $T(w, \lceil m/2 \rceil, s_B + w)$. Now, using the expression from (5.5), the arithmetic circuits that compute all the $E_{Y_A}(c_{j,\alpha})$ and all the $E_{Y_B}(g^\alpha)$ can be put together to obtain a circuit C' that computes $E_Y(f_j)$ for each $1 \leq j \leq w$. The size of C' can be bounded above as follows:

$$\begin{aligned}
 \text{size}(C') &\leq \sum_{j=1}^w \sum_{\alpha \in S} \text{size to compute } E_{Y_A}(c_{j,\alpha}) \\
 &\quad + \sum_{\alpha \in S} \text{size to compute } E_{Y_B}(g^\alpha) + \sum_{j=1}^w \sum_{\alpha \in S} 2 \\
 &\leq w2^w T\left(w, \left\lfloor \frac{m}{2} \right\rfloor, ws_A\right) + 2^w T\left(w, \left\lceil \frac{m}{2} \right\rceil, s_B + w\right) + w2^{w+1} \\
 &\leq w2^w 3ws_A \left\lfloor \frac{m}{2} \right\rfloor^{cw} + 2^w 3(s - s_A + w) \left\lceil \frac{m}{2} \right\rceil^{cw} + w2^{w+1} \\
 &\quad \text{using induction and the fact that } s = s_A + s_B
 \end{aligned}$$

$$\begin{aligned}
&\leq w2^w 3w s_A \left(\frac{7m}{8}\right)^{cw} + 2^w 3(s - s_A) \left(\frac{7m}{8}\right)^{cw} \\
&\quad + 2^w 3w \left(\frac{7m}{8}\right)^{cw} + w2^{w+1} \\
&\quad \text{since } \lceil \frac{m}{2} \rceil \leq \lceil \frac{7m}{8} \rceil \text{ for } m \geq 2 \\
&\leq 2^{3w} 3s \left(\frac{7m}{8}\right)^{cw} + 2^{3w} 3 \left(\frac{7m}{8}\right)^{cw} + 2^{3w} \\
&\quad \text{combining the first two terms and using } w \leq 2^w \text{ for } w \geq 1 \\
&\leq 3sm^{cw} \left[8^w \left(\frac{7}{8}\right)^{cw} + 8^w \left(\frac{7}{8}\right)^{cw} + 8^w \left(\frac{7}{8}\right)^{cw} \right] \\
&\leq 3sm^{cw} \text{ provided } c \geq \frac{5}{3 - \log 7} \geq \frac{3w + \log 3}{w(3 - \log 7)}.
\end{aligned}$$

□

6. Skew formulas

In this section, we consider the expressive power of exponential sums of polynomials computed by skew formulas.

It is well known that the complexity class **NP** is equivalent to $\exists \cdot \mathbf{P}$ and in fact even to $\exists \cdot \mathbf{F}$, where **F** is the class of languages decided by uniform polynomial-size formulas (circuits with out-degree 1 at each non-leaf node). (It is known that **F** equals **NC**¹.) A similar result holds in the case of Valiant's algebraic complexity classes too. Valiant has shown that **VNP** = $\Sigma \cdot \mathbf{VF}$ (see [Bürgisser 2000](#); [Bürgisser et al. 1997](#)), and thus the polynomial g in the expression for **VNP** using [Definition 2.2](#) can be assumed to be computable by a formula of polynomial size and polynomial degree.

Noting that **VNP** is the class of polynomials which are projections of the permanent polynomial family, a natural question arises about the polynomials which are equivalent to the determinant polynomial. Since the determinant exactly characterizes the class of polynomials which are computable by skew arithmetic circuits ([Toda 1991](#)), the question one could ask is: can the determinant be written as an exponential sum of partial instantiations of a polynomial that can be computed by *skew formulas* of poly

size, VSkewF ? Recall that a circuit is said to be skew if every \times (or \wedge in the boolean case) gate has at most one child that is not a circuit input. Skew circuits are essentially equivalent to branching programs. Thus, one could ask the related question: since $\text{VP} \subseteq \Sigma \cdot \text{VP} = \Sigma \cdot \text{VF}$, can we show that $\text{VP}_{\text{skew}} \subseteq \Sigma \cdot \text{VSkewF}$?

We show in [Theorem 6.5](#) that this is not possible. We first give an equivalent characterization of VSkewF in terms of “sparse polynomials” ([Lemma 6.1](#)) placing it inside VAC^0 ([Corollary 6.2](#)), and then use it to show that $\Sigma \cdot \text{VSkewF}$ is in fact equal to VSkewF ([Theorem 6.4](#)). Recall that VAC^0 denotes the class of polynomial families computed by arithmetic circuits of polynomial size, unbounded fan-in, and constant depth.

LEMMA 6.1. (i) *Let $f \in \mathbb{K}[X]$ be computed by a skew formula Φ of size s . Then the degree and number of monomials in f are bounded by s .*

(ii) *Conversely, if $f \in \mathbb{K}[X]$ is a degree d polynomial, where at most t monomials have non-zero coefficients, then f can be computed by a skew formula Φ of size $O(td)$.*

PROOF. Let F be a skew formula of size s . Consider a sub-tree T of F such that root of F is in T and for any gate g in T , if g is a $+$ gate then exactly one child of g is in T and if g is a \times gate then both children of g are present in T . We call such a subtree T a “proving subtree” of F . Since F is skew, T looks like a path, with edges hanging out at nodes labeled \times . But in a tree, the number of root to leaf paths is bounded by the number of leaves in the tree. Thus, the number of distinct proving subtrees of F is upper bounded by s . Let $p_F \in \mathbb{K}[X]$ be the polynomial computed by the formula F , where X is the set of input variables of F . It is easy to see that a proving subtree in F corresponds to a monomial in p_F (monomial with some value from \mathbb{K} as coefficient). Thus, the number of non-zero monomials in p_F is bounded by s . Since the degree of the monomial contributed by such a path is at most the length of the path, the degree of p_F is at most s .

On the other hand, if a polynomial $p \in \mathbb{K}[X]$ has t non-zero monomials m_1, \dots, m_t , then we can explicitly multiply variables

to get each monomial m_i and finally get the sum $\sum_i c_i m_i$, where $c_i \in \mathbb{K}$ is the coefficient of m_i in p . This formula computes p in size $O(td)$. \square

Note that the above translation from a degree d polynomial to a constant-depth circuit requires \times gates with fan-in d . Thus, unless $d \in O(1)$, the circuit is not an SAC^0 circuit. Here, SAC denotes semi-unbounded circuits, where the \vee or $+$ gates are allowed unbounded fan-in but the \wedge or \times gates are restricted to have constant fan-in.

COROLLARY 6.2. $\text{VSAC}^0 \subset \text{VSkewF} \subset \text{VAC}^0$.

PROOF. Polynomials computed by a VSAC^0 circuit have $O(1)$ degree. Thus, they have at most polynomially many distinct monomials, and hence from [Lemma 6.1.\(2\)](#) they are in VSkewF . From [Lemma 6.1.\(1\)](#), polynomials in VSkewF have $\text{poly}(n)$ monomials of $\text{poly}(n)$ degree, so they can be computed by a circuit of depth 2 and size $O(s^2)$, that is, in VAC^0 .

To see why the containments are proper: (1) The monomial $\prod_{i=1}^n x_i$ is in VSkewF , but its degree is not $O(1)$, and so it is not in VSAC^0 . (2) The function $\prod_{i=1}^n (x_i + y_i)$ is in VAC^0 but not in VSkewF because it has too many monomials. \square

REMARK 6.3. The constructions in [Lemma 6.1](#) work even in the multilinear world and allow us to construct equivalent syntactically multilinear skew formulas. Since the functions $\prod_{i=1}^n x_i$ and $\prod_{i=1}^n (x_i + y_i)$ have syntactically multilinear VSkewF formulas and VAC^0 circuits respectively, [Corollary 6.2](#) also holds if each of the classes referred to there is further restricted to be syntactically multilinear: $\text{sm-SAC}^0 \subset \text{sm-VSkewF} \subset \text{sm-AC}^0$.

THEOREM 6.4. $\Sigma \cdot \text{VSkewF} = \text{VSkewF}$.

PROOF. The containment $\text{VSkewF} \subseteq \Sigma \cdot \text{VSkewF}$ is obvious.

To show the converse, let $f \in \mathbb{K}[X]$ be such that $f(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where ϕ has a poly size skew formula and $m \leq \text{poly}(n)$. We need to show $f \in \text{VSkewF}$.

Since $\phi(X, Y)$ (where $X = x_1, \dots, x_n$ and $Y = y_1, \dots, y_m$) has a poly size skew formula, by [Lemma 6.1](#), we know that the number of non-zero monomials in ϕ is bounded by some polynomial $q(n, m)$, or equivalently, by some other polynomial $p(n)$. Hence, the number of non-zero monomials in $\phi(X, Y)|_X$ (*i.e.* monomials in X with coefficients from $\mathbb{K}[Y]$) is also bounded by $p(n)$. Summing these coefficient polynomials over all Boolean settings to Y yields the coefficients of f . Thus, f has no more than $p(n)$ monomials. The result now follows from [Lemma 6.1](#). \square

Since the number of monomials in the determinant polynomial is exponential, using [Lemma 6.1](#) and [Theorem 6.4](#) we obtain

THEOREM 6.5. *The family of determinant polynomials is not contained in the class $\Sigma \cdot \text{VSkewF}$.*

7. Conclusion and open questions

Summary of results:

1. We proposed a notion of “small space” for algebraic computations in terms of circuit width. **VL** was defined as the class of polynomials computed by log-width circuits with certain degree and constraints on the coefficients. However, it is easy to see that our definition of $\text{VWIDTH}(S(n))$ can be extended to polynomials with arbitrary coefficients from \mathbb{K} .

Since **VBP** is a natural arithmetic version of **NL**, a good validation of our thesis that width defines space would be a proof that $\text{VL} \subseteq \text{VBP}$. Unfortunately, we were unable to show any upper bound for **VL** better than the obvious bound **VP**.

2. We introduced the notion of read-once certificates and read-once exponential sums of arithmetic circuits. It is shown that with this definition, some classes behave on the expected lines:
 - 1) ABPs are closed under taking read-once exponential sums.
 - 2) Applying read-once exponential sum to **VP** yields exactly the class **VNP**.

For the case of $\Sigma^R \cdot \text{VL}$, the best upper bound we can see is only **VNP** which is obvious from the definition itself. However,

we believe that VL is indeed the analog of log-space and that read-once exponential sums over it does yield the analog of non-deterministic log-space. That is, we conjecture the following.

CONJECTURE 7.1. $\Sigma^R \cdot \text{VL} = \text{VBP}$.

Open questions We conclude with the following questions:

- Is VL contained in VBP ? *i.e.* do the class of all log width poly degree and size circuits have equivalent poly size algebraic branching programs?
- Is $\Sigma^R \cdot \text{VL} \subseteq \text{VP}$? Even in the case of VSC^0 , it will be interesting to see an upper bound of VP , *i.e.* is $\Sigma^R \cdot \text{VSC}^0 \subseteq \text{VP}$?
- Circuits corresponding to VSC families compute an output polynomial of polynomial degree, but intermediate nodes could compute polynomials of higher degree, with the high degree terms eventually canceling out. Such cancellations cannot always be made explicit without increasing the circuit width a lot; see [Limaye et al. \(2010\)](#) for a treatment of this. Therefore, we can define a further subclass of VP as polynomial families in VSC where the witnessing SC circuits have polynomial *syntactic degree*. (Syntactic degree is defined as follows: The syntactic degree of a leaf is 1 that of a $+$ or \vee node is the maximum of the syntactic degrees of its children and that of a \times or \wedge node is the sum of the syntactic degrees of its children. The syntactic degree of a circuit is the syntactic degree of its output node and is an upper bound on the degree of the computed polynomial.) sSC , or small SC , denotes polynomial-size poly-log width circuits with polynomial syntactic degree, and VsSC denotes families of polynomials computed by arithmetic circuits of this form.

Branching programs obey the syntactic small degree restriction. Thus, $\text{VBWP} \subseteq \text{VsSC}^0$. Since $\text{VNC}^1 = \text{VBWP}$, VNC^1 is contained in VsSC^0 . Is this containment proper? Are VNC^1 and VsSC^0 separate?

Can the study of read-once exponential sums throw some light on this question?

Since we do not have a nice definition of read-once certificates for depth bounded circuits, we use the equivalence $\text{VBWBP} = \text{VNC}^1$ for this purpose. From [Theorem 4.8](#), we have $\Sigma^R \cdot \text{VBWBP} = \text{VBWBP}$; hence, we can say that $\Sigma^R \cdot \text{VNC}^1 = \text{VNC}^1$. On the other hand, we do not know any upper bound for $\Sigma^R \cdot \text{VsSC}^0$ better than $\Sigma^R \cdot \text{VL} \subseteq \text{VNP}$. (In [Mahajan & Rao \(2009\)](#) it was erroneously claimed that $\Sigma^R \cdot \text{VsSC} \subseteq \text{VQP}$.)

- Is there any natural family of polynomials complete for VL?

Acknowledgements

The authors are indebted to the anonymous reviewers for very detailed comments, which helped to significantly improve the presentation of the results, and for pointing out a flaw in an earlier version. The results in this paper were announced in FCT 2009, in [Mahajan & Rao \(2009\)](#).

Much of this work was done when the second author was at The Institute of Mathematical Sciences, Chennai.

8. Appendix

8.1. Blum Shub Smale (BSS) model of computation. In this section, we briefly describe the model of computation over reals proposed by Blum, Shub and Smale. For more details, reader is referred to [Blum *et al.* \(1997\)](#). The BSS model is defined for computation over the field \mathbb{R} of real numbers. We present the version used in [de Naurois \(2006\)](#).

A BSS machine M has an input tape output tape and a work tape, where each cell stores a value from \mathbb{R} and a set of parameters $\mathcal{A} = \{A_1, \dots, A_k\}$, where $A_i \in \mathbb{R}$. In a single step, M can perform one of the following operations:

- *Input:* reads a value from the input tape into its work tape.
- *Computation:* Performs an arithmetic operation over values in the work tape (The number of operands is some fixed constant).

- *Output*: Writes a value on the output tape.
- *Constant*: Writes a constant $A_i \in \mathbb{R}$.
- *Branch*: Compares two real values and branches accordingly

Naturally, we can associate a function $\phi_M : \mathbb{R}^* \rightarrow \mathbb{R}$ with M . We say that a real set $L \subseteq \mathbb{R}^*$ is decided by M if the characteristic function of L , χ_L equals ϕ_M . We can make the machine M above non-deterministic by allowing non-deterministic choices at every step. $\mathsf{P}_{\mathbb{R}}$ is the set of all languages from \mathbb{R}^* that are decidable by polynomial time bounded BSS machines. Also, $\mathsf{NP}_{\mathbb{R}}$ is the class of languages that are computable by non-deterministic polynomial time bounded BSS machines.

In the unit space model, we count the number of work tape cells used by the machine as the space used. [Michaux \(1989\)](#) showed that any language that can be decided by a machine in the BSS model can in fact be decided using $O(1)$ space; that is,

PROPOSITION 8.1 ([Michaux \(1989\)](#)). *Let $L \subseteq \mathbb{R}$ be a language computed by a machine M . Then there is machine M' and a constant k such that M' computes L using space k .*

References

- SANJEEV ARORA & BOAZ BARAK (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA. ISBN 0521424267, 9780521424264.
- VIKRAMAN ARVIND, PUSHKAR S. JOGLEKAR & SRIKANTH SRINIVASAN (2010). On Lower Bounds for Constant Width Arithmetic Circuits. In *ISAAC*, YINGFEI DONG, DING-ZHU DU & OSCAR IBARRA, editors, Lecture Notes in Computer Science, 637–646. Springer Berlin/Heidelberg.
- DAVID A. MIX BARRINGTON (1989). Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1 . *Journal of Computer and System Sciences* **38**(1), 150–164.
- LENORE BLUM, FELIPE CUCKER, MIKE SHUB & STEVE SMALE (1997). *Complexity and Real Computation*. Springer.

ALLAN BORODIN (1977). On Relating Time and Space to Size and Depth. *SIAM J. Comput.* **6**(4), 733–744.

P. BÜRGISSER, M. CLAUSEN & M. A. SHOKROLLAHI (1997). *Algebraic Complexity Theory*. Springer-Verlag.

PETER BÜRGISSER (2000). *Completeness and Reduction in Algebraic Complexity Theory*. Algorithms and Computation in Mathematics. Springer-Verlag.

HERVÉ CAUSSINUS, PIERRE MCKENZIE, DENIS THÉRIEN & HERIBERT VOLLMER (1998). Nondeterministic NC^1 Computation. *Journal of Computer and System Sciences* **57**, 200–212.

A. CHIU, G. DAVIDA & B. LITOW (2001). Division in Logspace-Uniform NC^1 . *RAIRO Theoretical Informatics and Applications* **35**, 259–276.

S. COOK (1971). Characterizations of pushdown machines in terms of time-bounded computers. *Journal of Association for Computing Machinery* **18**, 4–18.

STEPHEN A. COOK (1979). Deterministic CFL's Are Accepted Simultaneously in Polynomial Time and Log Squared Space. In *Proceedings of the ACM Symposium on Theory of Computing STOC*, 338–345.

UFFE FLARUP & LAURENT LYAUDET (2010). On the Expressive Power of Permanents and Perfect Matchings of Matrices of Bounded Pathwidth/Cliquewidth. *Theory Comput. Syst.* **46**(4), 761–791.

RAYMOND GREENLAW, JAMES HOOVER & WALTER RUZZO (1995). *Limits To Parallel computation: P-Completeness Theory*. Oxford University Press.

MAURICE JANSEN & JAYALAL M. N. SARMA (2010). Balancing Bounded Treewidth Circuits. In *CSR*, volume 6072 of *Lecture Notes in Computer Science*, 228–239.

MAURICE J. JANSEN (2008). Lower Bounds for Syntactically Multilinear Algebraic Branching Programs. In *MFCS*, 407–418.

MAURICE J. JANSEN & B. V. RAGHAVENDRA RAO (2009). Simulation of Arithmetical Circuits by Branching Programs with Preservation of Constant Width and Syntactic Multilinearity. In *CSR*, 179–190.

DAVID S. JOHNSON (1990). A Catalog of Complexity Classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, JAN VAN LEEUWEN, editor, 67–161. Elsevier and MIT Press.

ERICH KALTOFEN & PASCAL KOIRAN (2008). Expressing a fraction of two determinants as a determinant. In *ISSAC*, 141–146.

PASCAL KOIRAN & SYLVAIN PERIFEL (2009a). VPSPACE and a transfer theorem over the complex field. *Theor. Comput. Sci.* **410**(50), 5244–5251.

PASCAL KOIRAN & SYLVAIN PERIFEL (2009b). VPSPACE and a Transfer Theorem over the Reals. *Computational Complexity* **18**(4), 551–575.

NUTAN LIMAYE, MEENA MAHAJAN & B. V. RAGHAVENDRA RAO (2010). Arithmetizing Classes Around NC^1 and L. *Theory of Computing Systems* **46**(3), 499–522. Preliminary version in STACS 2007, LNCS vol. 4393 pp. 477–488.

MEENA MAHAJAN & B. V. RAGHAVENDRA RAO (2009). Small-Space Analogues of Valiant’s Classes. In *17th International Symposium on Fundamentals of Computation Theory, FCT*, 250–261.

GUILLAUME MALOD (2007). The Complexity of Polynomials and Their Coefficient Functions. In *IEEE Conference on Computational Complexity*, 193–204.

GUILLAUME MALOD & NATACHA PORTIER (2008). Characterizing Valiant’s algebraic complexity classes. *J. Complexity* **24**(1), 16–38.

CHRISTIAN MICHAUX (1989). Une remarque à propos des machines sur \mathbb{R} introduites par Blum, Shub et Smale. *Comptes Rendus de l’Académie des Sciences de Paris* **309**(7), 435–437.

PAULIN JACOBÉ DE NAUROIS (2006). A Measure of Space for Computing over the Reals. In *CiE*, 231–240.

NOAM NISAN & AVI WIGDERSON (1995). On the complexity of bilinear forms. In *STOC ’95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, 723–732. ACM, New York, NY, USA.

NICHOLAS PIPPENGER (1979). On Simultaneous Resource Bounds. In *20th Annual Symposium on Foundations of Computer Science FOCS*, 307–311.

AMIR SHPILKA & AVI WIGDERSON (2002). Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity* **10**, 1–27. ISSN 1016-3328.

SEINOSUKE TODA (1991). Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. Comp. Sci. and Inf. Math., Univ. of Electro-Communications, Tokyo.

SEINOSUKE TODA (1992). Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Informations and Systems* **E75-D**, 116–124.

IDDO TZAMARET (2008). *Studies in Algebraic and Propositional Proof Complexity*. Ph.D. thesis, Tel Aviv University.

L. G. VALIANT (1979a). Completeness classes in algebra. In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, 249–261. ACM, New York, NY, USA.

LESLIE G. VALIANT (1976). Graph-theoretic properties in Computational Complexity. *Journal of Computer and System Sciences* **13**, 278–285.

LESLIE G. VALIANT (1979b). The Complexity of Computing the Permanent. *Theor. Comput. Sci.* **8**, 189–201.

LESLIE G. VALIANT (1982). Reducibility by algebraic projections. *Logic and Algorithmic: an International Symposium held in honour of Ernst Specker* **30**, 365–380.

H. VENKATESWARAN (1992). Circuit Definitions of Nondeterministic Complexity Classes. *SIAM Journal on Computing* **21**, 655–670.

V VINAY (1991). Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of 6th Structure in Complexity Theory Conference*, 270–284.

H. VOLLMER (1999). *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York Inc.

Manuscript received 18 June 2010

MEENA MAHAJAN
The Institute of Mathematical
Sciences,
Chennai 600113,
India.
`meena@imsc.res.in`
`http://www.imsc.res.in/~meena/`

B. V. RAGHAVENDRA RAO
Universität des Saarlandes,
Informatik,
66041 Saarbrücken,
Germany.
`bvrr@cs.uni-sb.de`
`http://www.imsc.res.in/~bvrr/`