Small space analogues of Valiant's classes and the limitations of skew formula

Meena Mahajan
The Institute of Mathematical Sciences,
Chennai, India
meena@imsc.res.in

B. V. Raghavendra Rao*
Universität des Saarlandes, Informatik
66041 Saarbrücken, Germany
bvrr@cs.uni-sb.de

December 22, 2009

Abstract

In the uniform circuit model of computation, the width of a boolean circuit exactly characterises 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. We introduce the class VL as an algebraic variant of deterministic log-space L. In the uniform setting, we show that our definition coincides with that of VPSPACE at polynomial width.

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 can be expressed as a read-once exponential sum over polynomials in VL, *i.e.* $\mathsf{VBP} \in \Sigma^R \cdot \mathsf{VL}$. We also show that $\Sigma^R \cdot \mathsf{VBP} = \mathsf{VBP}$, *i.e.* VBPs are stable under read-once exponential sums. Further, we show that read-once exponential sums over a restricted class of constant-width arithmetic circuits are within VQP, and this is the largest known such subclass of poly-log-width circuits with this property.

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

1 Introduction

Apart from characterizing algebraic computations, an interesting task in algebraic complexity theory is to define algebraic complexity classes analogous to the ones of boolean complexity theory. The classes VP,VNP, VNC, VSC, VsSC and VBP have P,NP, NC, SC, sSC and BP respectively as their boolean counterparts. Perhaps, the prominent boolean complexity classes that do not have algebraic analogues are the space complexity classes.

*This work was done when this author was at the Institute of Mathematical Sciences.

Dagstuhl Seminar Proceedings 09421
Algebraic Methods in Computational Complexity
http://drops.dagstuhl.de/opus/volltexte/2010/2412

The main obstacle in this direction is defining a "right" measure for space. Two obvious choices are: 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, [Val76, NW95, Mic89, dN06, KP07b, KP07a], with varying degrees of success. In particular, the models of [Mic89, KP07b, KP07a] when adapted to logarithmic space are too powerful to give meaningful insights into small-space classes, whereas the model of [dN06] 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, and captures both succinctness of coefficients and ease of evaluating the polynomials. We show that our notion of space $\mathsf{VSPACE}(s)$ coincides with that of [KP07b, KP07a] at polynomial space with uniformity (Theorem 7), and so far avoids the pitfalls of being too powerful or too weak at logarithmic space.

Continuing along this approach, we propose a way of describing non-deterministic space-bounded computation in this context. The specific motivation for this is to obtain an algebraic analogue of the class non-deterministic log-space NL as well as an analogue of the result that $\mathsf{VNP} = \Sigma \cdot \mathsf{VP}$. Again, there is a well-known model for NL that easily carries over to the arithmetic setting, namely polynomial-size branching programs BP. But we are unable to compare VBP even with our version of VL. Our model here for NL is based on read-once certificates, which also provide the correct description of NL in terms of L in the Boolean world. We show that the arithmetization of this model, $\Sigma^R \cdot \mathsf{VL}$, does contain arithmetic branching programs (Theorem 14).

Surprisingly, we are unable to show the converse. In fact, we are unable to show a good 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 18). Similarly, for polylog-width circuits where the syntactic degree is bounded by a polynomial, read-once certification does not take us beyond VQP (Theorem 21). Further, if the circuit is multiplicatively disjoint and of constant width, then read-once certification does not take us beyond VP.

We also study the class of polynomial size skew formulas, denoted SkewF. The motivation for this study arises from Valiant's characterisations of the classes VP and VNP (see [Val79]). Valiant proved that every polynomial $p(X) \in \mathsf{VNP}_{\mathbb{K}}$ (where $\mathbb{K}$ is an arbitrary ring), and in particular every polynomial in $\mathsf{VP}_{\mathbb{K}}$, can be written as $p(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where the polynomial ϕ has an arithmetic formula of polynomial size. We know that the class of "Permanent" (see e.g.[Bür00]) polynomials is complete for VNP. It is also known [Tod91] that the class "Determinant" is equivalent to 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 formulae? 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, NC, L, NL and LogCFL (see e.g. [Vol99],[AB09]).

An arithmetic circuit over a ring $\langle \mathbb{K}, +, \times, 0, 1\rangle$ is a directed acyclic graph C, where vertices with non-zero in-degree are labelled from $\{+, \times\}$, and vertices of zero-in-degree (called *leaf* nodes) are labelled from $X \cup \mathbb{K}$, where $X = \{x_1 \ldots, x_n\}$ 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 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, its *size* is the number of nodes and edges in it, and its *width*, if it is a layered circuit, is the maximum number of nodes at any particular layer. We assume that all output nodes appear at the last layer.

Polynomial size poly-log depth Boolean circuits form the class NC; NC^1 is the subclass of log-depth circuits and is known to be contained in L. Polynomial size poly-log width boolean circuits 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 ([Bar89]) and uniform SC^1 equals L.

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 in $X \cup \mathbb{K}$. C is said to be *weakly skew* if for every $f = g \times h$, either the edge (g, f) or (h, f) is a bridge in the circuit, i.e removing the edge disconnects its end-points in the resulting the circuit. Poly-size Boolean skew circuits are known to characterise NL ([Ven92]).

An *algebraic branching program* (BP for short) over a ring $\mathbb{K}$ is a layered directed acyclic graph, where edges are labelled from $\{x_1, \ldots, x_n\} \cup \mathbb{K}$. 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 number of layers in it. The depth of a BP B equals $1 + \mathsf{length}(B)$. The polynomial P computed by a BP is the sum of weights of all s-t paths in P, where 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 generalised in the obvious way to several nodes $t_1, t_2, \ldots, t_m$ existing at the last level. Note that BPs can be simulated by skew circuits and vice versa with a constant blow up in the width.

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

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

VP_e is the sub-class of VP corresponding to poly-size arithmetic formula (*i.e.* circuits with out-degree at most 1, also called *expressions*). If f_n can be computed by arithmetic circuits with the resource bounds the same as NC^1 or SC^1 or SC^1, then we say the family is in VNC^1 or VSC^0 or VSC^1 respectively. It is known that VP_e is the same as VNC^1. If the circuits computing f_n have quasipolynomial size $2^{\log^c n}$, we say that $\{f_n\}$ is in the class VQP.

A polynomial family $(f_n)_{n\geq 0}$ is in VNP if there exists a family $(g_\ell)_{n\geq 0}$ in VP such that $f_n(X) = \sum_{e\in\{0,1\}^m} g_n(X, e)$, where m is bounded by $\mathsf{poly}(n)$.

We let VBP and VBWBP stand for classes corresponding to *poly*-size BPs of *poly* and *constant* width, respectively. Without loss of generality, we can treat these classes as skew circuits. ([MP06])

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 [Vol99])

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). In the case of arithmetic computations, defining a notion of "space bounded computation" seems to be a a hard task.

3.1 Previously studied notions

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 way back in 1976 and later by Nisan and Wigderson ([Val76, NW95]). This was in the context of time-space trade-offs for arithmetic straight-line programs. Subsequently, Michaux [Mic89] showed that with this notion of space, any language that is decided by a machine in the Blum-Shub-Smale model of computation (a general model for algebraic computation capturing the idea of computation over reals, [BCSS97]; see also [Bür00]) can also be computed using $O(1)$ registers. Hence there is no space-hierarchy theorem under this space measure.

Another possible measure is the size of a succinct description of the polynomials computed at each cell. In [dN06], Naurois introduced a notion of weak space in the Blum-Shub-Smale model, and introduced the corresponding log space classes $\mathsf{LOGSPACE}_W$and

PSPACE_W. This is in fact 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. $\mathsf{LOGSPACE}_W$ seems to be too weak to contain even NC^1 over $\mathbb{R}$, which is in contrast to 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 the 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 formula (see Section 6), which is in fact a proper subclass of constant depth arithmetic circuits (i.e., VAC^0).

Koiran and Perifel ([KP07b, KP07a]) suggested another notion of polynomial space for Valiant's ([Val79, Bür00]) classes. The main purpose of their definition was to prove a transfer theorem over $\mathbb{R}$ and $\mathbb{C}$. Under their definition Uniform-VPSPACE (the non-uniform counterpart can be defined similarly) is defined as the set of families (f_n) of multivariate polynomials $f_n \in F[x_1, \ldots, x_{u(n)}]$ with integer coefficients such that

- $u(n)$ is bounded by a polynomial in n.
- Size of coefficients of f_n is bounded by $2^{poly(n)}$.
- Degree of f_n is bounded by $2^{poly(n)}$.
- Every bit of the coefficient function of f_n is computable in PSPACE.

In [KP07b], it was observed that the class VPSPACE is equivalent to the class of polynomials computed by arithmetic circuits of polynomial depth and exponential size. Such Boolean circuits 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 $\mathsf{VSPACE}(s(n))$ to consist of families $(f_n)_{n\geq 1}$ of polynomials satisfying the following:

- $f \in \mathbb{Z}[x_1, \ldots, x_{u(n)}]$, where $u(n)$, the number of variables in f_n, is bounded by some polynomial in n.
- 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 ith bit of the coefficient of M in f_n is computable in $\mathsf{DSPACE}(s(n))$.

It is easy to see that with this definition, even the permanent function PERM_n is in log-space. Thus $\mathsf{VSPACE}(\log n)$ would be too big a class to be an arithmetic version of log-space. The reason here is that this definition, unlike that of [dN06], 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 [Mal07].

3.2 Defining VPSPACE 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.

Definition 1 *Let* $\mathsf{VWIDTH}(S)$ *(with* $S = S(n)$*) be 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* $\mathsf{poly}(n)$
- $f_n \in \mathbb{Z}[x_1, \ldots, x_{u(n)}]$*, i.e* f_n *has only integer coefficients*
- $\deg(f) \leq \max\{2^{S(n)}, \mathsf{poly}(n)\}$.
- *The coefficients of* f_n *are representable using* $\max\{2^{S(n)}, \mathsf{poly}(n)\}$ *many bits.*
- f_n *is computable by an arithmetic circuit of width* $S(n)$ *and size* $\max\{2^{S(n)}, \mathsf{poly}(n)\}$.

Further, if the arithmetic circuits in the last condition are DSPACE(S)*-uniform, we call the family Uniform-*$\mathsf{VWIDTH}(S)$.

Remark 2 *For* $i \geq 2$ $\mathsf{VWIDTH}(\log^i n)$ *is very close to* VSC^i *though they are different for the following reasons:*

- *Polynomials in* $\mathsf{VWIDTH}(\log^i n)$ *can have degree* $O(2^{\log^i n})$*, whereas degree of polynomials in* VSC^i *is bounded by* $\mathsf{poly}(n)$.
- *Coefficients of* $\mathsf{VWIDTH}(\log^i n)$ *are integers and their size is bounded by* $O(2^{\log^i n})$*, whereas* VSC^i *can have arbitrary coefficients.*

However, when $i = 0, 1$ *they coincide, if we restrict the polynomials in* VSC^0 *and* VSC^1 *to have only integer coefficients. Also, it is easy to see that* $\mathsf{VWIDTH}(\log^i n)$ *and* VsSC^i *are different. Apart from the above two, another major difference between them is:*

- *The circuits in* $\mathsf{VWIDTH}(\log^i n)$ *can have super-polynomial syntactic degree.*

We show in Theorem 7 below that with this definition, uniform $\mathsf{VWIDTH}(\mathsf{poly})$ coincides with uniform $\mathsf{VPSPACE}$ as defined in [KP07b]; thus polynomial width indeed corresponds to polynomial space. Motivated by this equivalence, we define the following complexity classes:

Definition 3 $\mathsf{VSPACE}(S(n)) = \mathsf{VWIDTH}(S(n))$
Uniform-$\mathsf{VSPACE}(S(n))=$ *Uniform*-$\mathsf{VWIDTH}(S(n))$

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

The following containments and equalities follow directly from known results (from [CMTV98] and [LMR07]) about width-constrained arithmetic circuits.

Lemma 4 $\mathsf{VBWBP} = \mathsf{VNC}^1 = \mathsf{VP}_e \subseteq \mathsf{VSPACE}(O(1)) = \mathsf{VSC}^0 \subseteq \mathsf{VL} = \mathsf{VSC}^1 \subseteq \mathsf{VP}$

Remark 5 *In the above equivalence, we need to extend* $\mathsf{VWIDTH}(S(n))$ *to include arbitrary constants from* $\mathbb{K}$*. In this case we omit the bound on the size of the coefficients in definition 1.*

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 6 *For every* $S(n) > \log n$*, the classes* $\mathsf{VSPACE}(S(n))$ *are closed under polynomially bounded summations and constant many products.*

3.3 Comparing VPSPACE and VWIDTH(poly)

This subsection is devoted to proving the following equivalence,

Theorem 7 *The class Uniform-*$\mathsf{VPSPACE}$ *as defined in [KP07b] coincides with Uniform-*$\mathsf{VWIDTH}(\mathsf{poly})$*.*

We use the following easy fact:

Fact 8 *A* d *degree polynomial over* t *variables has at most* $\binom{d+t}{t}$ *monomials.*

Now, Theorem 7 follows from the two lemmas below.

Lemma 9 *Uniform-*$\mathsf{VPSPACE} \subseteq$ *Uniform-*$\mathsf{VWIDTH}(\mathsf{poly})$*.*

Proof: Let $(f_n)_{n\geq 0}$ be a family of polynomials in $\mathsf{VPSPACE}$. Then by definition, the bits of the coefficients of f_n can be computed in PSPACE and hence by exponential size polynomial width circuits. The (exponentially many) bits can be put together with appropriate weights to obtain a 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 8, there are only exponentially many distinct monomials. Thus we get a polynomial width exponential-size circuit computing f_n. ■

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

Lemma 10 *Uniform*-$\mathsf{VWIDTH}(\mathsf{poly}) \subseteq$ *Uniform*-$\mathsf{VPSPACE}$.

Proof: Let $(f_n)_{n\geq 0}$ be a family of polynomials in $\mathsf{VWIDTH}(\mathsf{poly}(n))$. 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 $w(n) = \mathsf{poly}(n)$ and $s(n) \in 2^{O(n^c)}$ respectively be the width and size of a witnessing circuit C.

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

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

Using Lagrangian interpolation for multivariate polynomials we have

$$\begin{aligned} f(\tilde{x}) = \sum_{\tilde{i}\in T} f(\tilde{i})\mathsf{Equal}(\tilde{x}, \tilde{i}) \quad &= \quad \sum_{\tilde{i}\in T} f(\tilde{i}) \prod_{k=1}^{N} \mathsf{Equal}(x_k, i_k) \\ \text{where } \ \mathsf{Equal}(x, i) = \prod_{a\in S\setminus\{i\}} \left(\frac{x-a}{i-a}\right) \quad &= \quad \frac{\prod_{a\in S\setminus\{i\}}(x-a)}{i!(d-i)!(-1)^{d-i}} \end{aligned}$$

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

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

But we have a nice form for the inner numerator:

$$\mathsf{coeff} \text{ of } x_k^{t_k} \text{ in } \textstyle\prod_{a\in S\setminus\{i\}}(x_k - a) \text{ equals } (-1)^{d-t_k} S_{d,t_k}(0, 1, \ldots, i_k - 1, i_k + 1, \ldots, d)$$

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

To compute the desired coefficient in PSPACE, we use the Chinese Remaindering technique; See [CDL01] for more details. Since symmetric polynomials are easy to compute (*e.g.* [SW99] or Th 2.5.4 in [Tza08]), and since $f(\tilde{i})$ is computable by a polynomial-width 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 $f(\tilde{i}) \mod p$). Reconstructing the coefficient from its residues modulo all such primes can also be performed in PSPACE. ■

Lemma 10 requires that the VWIDTH family be uniform (with a direct-connection uniformity condition). If the VWIDTH family is non-uniform, this problem cannot be circumvented with polynomial advice, since the circuit has exponential size.

4 Read-Once certificates

In general, non-deterministic complexity classes can be defined via existential quantifiers. *e.g.* , $\mathsf{NP} = \exists \cdot \mathsf{P}$. 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.* , $\mathsf{VNP} = \Sigma \cdot \mathsf{P}$. It is also known that $\mathsf{VNP} = \Sigma \cdot \mathsf{VP}_e = \Sigma \cdot \mathsf{VNC}^1$ (see [Bür00]).

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 $\exists \cdot \mathsf{L} = \mathsf{NP}$. It is known that with the notion of "read once" certificates (see, *e.g.* , [AB09], 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 11 *Let C be a layered arithmetic circuit with ℓ layers. Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ 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, C satisfies the following:*

- *There is a fixed permutation $\pi \in S_m$ such that the variables of Y appear in the order $y_{\pi(1)}, \ldots, y_{\pi(m)}$ along any leaf-to-root path.*
- *There exist indices $0 = i_1 \leq \ldots \leq i_m \leq i_{m+1} = \ell$ such that the variable $y_{\pi(j)}$ appears only from layers $i_j + 1$ to i_{j+1}.*

We henceforth without loss of generality, assume that π is the identity permutation.

Now we define the the exponential sum over read-once certified circuits.

Definition 12 *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 $m(n) = n + m'(n)$, $m'(n) \leq \mathsf{poly}(n)$, $f_n(X) = \sum_{Y\in\{0,1\}^{m'(n)}} g_{m(n)}(X, Y)$ 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

Lemma 13 $\Sigma^R \cdot \mathsf{VP} = \Sigma \cdot \mathsf{VP} = \mathsf{VNP}$

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 $\mathsf{BP} = \mathsf{NL} = \Sigma^R \cdot \mathsf{L}$, we should be able to show that VBP equals $\Sigma^R \cdot \mathsf{VL}$. 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 14 $\mathsf{VBP} \subseteq \Sigma^R \cdot \mathsf{VL}$.

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

Definition 15 *A polynomial $f \in \mathbb{K}[X_1, \ldots, X_n]$ is called a projection of g (denoted $f \leq g$), if*

$$f(X_1, \ldots, X_n) = g(a_1, \ldots, a_m)$$

where each $a_i \in \mathbb{K} \cup \{X_1, \ldots, X_n\}$.

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

$$\exists n_0, \forall n \geq n_0,\ f_n \leq g_{m(n)}$$

where $m(n) \leq \mathsf{poly}(n)$.

Let $(G_n) = (V_n, E_n)$ (with $|V_n| = m = \mathsf{poly}(n)$) be a family of directed acyclic graphs and let $s = 1$ and $t = n$ denote two special nodes in G_n. We assume without loss of generality that the graph is topologically sorted; edges are from i to j only when $i < j$. Let $A = (a_{i,j})_{i,j\in\{1,\ldots,m\}}$ be an $m \times m$ matrix with variable entries, representing edge weights in G_n. For any directed $s-t$ path $P = \langle v_0, v_1, \ldots, v_\ell, v_{\ell+1}\rangle$ in G_n, let M_P denote the monomial that is the product of the variables corresponding to edges in P. Let $PATH_G^n = \sum_P M_P$, where P ranges over all the $s-t$ paths in G_n.

Definition 16

$$PATH = \left\{ (PATH_G^n)_{n\geq 0} \ \middle|\ \begin{array}{l} G{=}(G_n)_{n\geq 0} \textit{ is a family of layered complete di-} \\ \textit{rected acyclic graphs with edges of } G_n \textit{ labeled from} \\ \{x_1, \ldots, x_n\} \end{array} \right\}$$

It is easy to see the following:

Proposition 17 *(folklore) $PATH$ is complete for VBP under projections.*

We prove theorem 14 by showing that $PATH_G^n \in \Sigma^R \cdot \mathsf{VL}$ for any layered directed acyclic graph family $G = (G_n)_{n\geq 0}$.

Proof:[of theorem 14] Here onwards we drop the index n from G_n.

We define function $h_G(Y, Z) : \{0,1\}^{\lceil \log m \rceil} \times \{0,1\}^{m^2} \to \{0,1\}$ as follows. Assume that the variables in $Y = \{y_1, \ldots, y_k\}$ and $Z = \{z_{1,1}, \ldots, z_{m,m}\}$ take only values from $\{0,1\}$. $h_G(Y, Z) = 1$ if and only if $Z = z_{1,1}, \ldots, z_{m,m}$ represents a directed s-t path in G of length exactly ℓ, where ℓ written in binary is $y_1 \ldots y_k$.Note that s-t paths P in G are in one-to-one correspondence with assignments to Y, Z such that $h_G(Y, Z) = 1$. Hence

$$\begin{aligned} PATH_G^n = \sum_P M_P &= \sum_{Y,Z} h_G(Y,Z)\,[\text{ weight of path specified by } Y, Z] \\ &= \sum_{Y,Z} h_G(Y,Z) \prod_{i,j} (a_{i,j} z_{i,j} + (1 - z_{i,j})) \end{aligned}$$

There is a deterministic log-space algorithm A which computes $h_G(Y,Z)$ when Y,Z is given on a "read once" input tape (see [AB09]). Let C be the corresponding $O(\log n)$ width boolean circuit. (without loss of generality, assume that all negation gates in C are at the leaves.) Let D the natural arithmetization of C. Since Y and Z are on a read-once input tape, it is easy to see that C, and hence D, are read-once certified in the variables from Y and Z. We can attach, parallel to D, constant-width circuitry that collects factors of the product $\prod_{i,j}(a_{i,j}z_{i,j} + (1 - z_{i,j}))$ as and when the $z_{i,j}$ variables are read, and finally multiplies this with $h_G(Y,Z)$. The resulting circuit remains $O(\log n)$-width, and remains read-once certified on Y, Z. ■

While we are unable to show the converse, we are also unable to show a reasonable upper bound on $\Sigma^R\cdot\mathsf{VL}$. It is not even clear if $\Sigma^R\cdot\mathsf{VL}$ 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 18 *1.* $\Sigma^R \cdot \mathsf{VBP} = \mathsf{VBP}$

2. $\Sigma^R \cdot \mathsf{VBWBP} = \mathsf{VBWBP}$

This theorem follows from Lemma 20. We need the following notation:

Definition 19 *For $f \in \mathbb{K}[X,Y]$ with $X = \{x_1,\ldots,x_n\}$ and $Y = \{y_1,\ldots,y_m\}$, $E_Y(f)$ denotes the exponential sum of $f(X,Y)$ over all Boolean settings of Y. That is,*

$$E_Y(f)(X) = \sum_{e\subseteq\{0,1\}^m} f(X,e)$$

Lemma 20 *Let C be a layered skew arithmetic circuit on variables $X \cup Y$. Suppose C is* read-once *certified in Y. Let $w = \mathsf{width}(C)$, $s = \mathsf{size}(C)$ and $\ell =$ the number of layers in C. Let $f_1,\ldots,f_w$ denote the output gates (also the polynomials computed by them) of C. There exists a weakly skew circuit C', of size $O(mw^4s)$ and width $4w$, that computes all the exponential sums $E_Y(f_1),\ldots,E_Y(f_w)$.*

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 corresponding outputs we get a circuit C' which computes the required function. Clearly $\mathsf{width}(C') \leq 2w$ and $\mathsf{size}(C') \leq 3s + w$.

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 $Y \setminus \{y_m\} = \{y_1,\ldots,y_{m-1}\}$. As per definition 11, the layers of C can be partitioned into m blocks, with the kth block reading only y_k from Y. Let $0 = i_1 \leq i_2 \leq \ldots \leq i_m \leq \ell$ be the layer indices such that y_k is read between layers $i_k + 1$ and i_{k+1}. Let $f_1,\ldots,f_w$ 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 $g_1,\ldots,g_w$ be the output gates of D. These are also the inputs to C_m; we symbolically relabel the non-leaf inputs at level 0 and the outputs of C_m as $Z_1, \ldots Z_w$ and $h_1, \ldots, h_w$. Clearly, C_m and D are both skew circuits of width w. Further, each h_j depends on X, y_m and Z; that is, $h_1, \ldots, h_w \in R[Z_1, \ldots, Z_w]$ where $R = \mathbb{K}[X, y_m]$. Similarly, each g_j depends on X and Y'; $g_1, \ldots, g_w \in \mathbb{K}[X, Y']$. The values computed by C can be expressed as $f_j(X, Y) = h_j\left(X, y_m, g_1(X, Y'), \ldots, g_w(X, Y')\right)$.

Since C and C_m are skew circuits, and since the variables Z_j represent non-leaf gates of C, C_m must be 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

$$\begin{aligned} f_j(X, Y) &= h_j\left(X, y_m, g_1(X, Y'), \ldots, g_w(X, Y')\right) \\ &= c_j(X, y_m) + \sum_{k=1}^{w} c_{j,k}(X, y_m) g_k(X, Y') \qquad \text{and hence} \end{aligned}$$

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

$$= 2^{m-1} \sum_{e_m = 0}^{1} c_j(X, e_m) + \sum_{k=1}^{w} \left(\sum_{e_m \in \{0,1\}} c_{j,k}(X, e_m) \right) \left(\sum_{e' \in \{0,1\}^{m-1}} g_k(X, e') \right)$$

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

By induction, we know that there is a weakly skew circuit D' of width $4w$ and size $O((m-1)w^4 s)$ 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 Z_k replaced by 0 computes exactly $c_j(X, y_m)$. So the sum can be computed as in the base case, in width $w+1$ and size $3(\mathsf{size}(C_m)+1)$. Multiplying this by 2^{m-1} in the standard way adds nothing to width and 2 to size, so overall width is $w+1$ and size is at most $2s+4$.

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 $c_j(X, y_m)$ (as computed above) from this gives $c_{j,k}(X, y_m)$. Now, the sum can be computed as in the base case. Again, to compute $E_{y_m}(c_{j,k})(X)$, we use two copies of the difference circuit with $y_m = 0$ and $y_m = 1$ hardwired, and add their outputs. It is easy to see that this circuit has width $w+2$ and size at most $4(w+2)\mathsf{size}(C_m) \leq 4(w+2)s$.

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 $E_Y(f_j)$, $w+2$ for computing the terms in the sum above (they are computed sequentially so the width does not add up), and 2 for computing partial sums in this process, overall at most $3w+4$. Thus the resulting circuit has width at most $\max\{\mathsf{width}(D'), 3w+4\} \leq 4w$.

To bound the size of the circuit, we bound its depth in the part above D' by d; then size is at most $\mathsf{size}(D') + \mathsf{width} \times d$. The circuit has w modules to compute the $E_Y(f_j)$s. The depth of each module can be bounded by the depth 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 $(2s+4) + w \times 4(w+2)s$. So $d \leq w(2s+4+4sw(w+2)) = \theta(w^3 s)$, and the size bound follows. ■

Now we prove Theorem 18:

Proof:[of Theorem 18]

(1) Since weakly skew circuits can be transformed into skew circuits ([MP06]) with a constant blowup in the size([Jan08]), the equivalence follows.

(2) From [JR09], we know that when width of the weakly skew circuit is a constant, width of the resulting skew circuit will again be a constant. *i.e.* the resulting circuit now will have width $O(w^2)$ and size $O(w 2^{4w} m w^4 s)$. ■

5 Read-Once exponential sums of some restricted circuits

In this section, we explore how far the result of Theorem 18 can be pushed to larger classes within VP. In effect, we ask whether the technique of Lemma 20 is applicable to larger classes of circuits. Such a question is relevant because we do not have any bound (better than VNP) even for $\Sigma^R \cdot \mathsf{VSC}^0$ and $\Sigma^R \cdot \mathsf{VL}$.

One 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.

A further generalization we consider is polynomial syntactic degree bounded arithmetic circuits.

Examining the proof of Lemma 20, we see that the main barrier in extending it to these larger classes is that when we slice C into D and C_m, C_m is no longer 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 21 *1.* $\Sigma^R \cdot md\text{-}\mathsf{VSC}^0 \subseteq \mathsf{VP}$.

2. $\Sigma^R \cdot md\text{-}\mathsf{VSC} \subseteq \mathsf{VQP}$.

3. For all $i \geq 0$, $\Sigma^R \cdot \mathsf{VsSC}^i \subseteq \mathsf{VQP}$.

Proof strategy for Theorem 21.

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 equation 1 below to obtain the required $E_Y(f_j)$s.

This strategy is spelt out in detail for the case of multiplicative disjoint circuits in Lemma 23. For syntactic degree bounded by a polynomial, Step 3 above needs special treatment, spelt out in Lemma 25.

We need the following observation (which is already used implicitly in the proof of Lemma 20):

Observation 22 *1. If* $f = g + h$, *then* $E_Y(f) = E_Y(g) + E_Y(h)$.

2. 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*

$$E_Y(f) = E_{Y_g}(g) \times E_{Y_h}(h) \tag{1}$$

Lemma 23 *Let* C *be a layered multiplicatively disjoint circuit of width* w *and size* s *on variables* $X \cup Y$. *Let* ℓ *be the number of layers in* C. *Suppose* C *is read-once certified in* Y. *Let* $f_1 \ldots, f_w$ *be the output gates of* C. *Then, there is an arithmetic circuit* C' *of size* $sm^{O(w)}$ *which computes* $E_Y(f_1), \ldots, E_Y(f_w)$.

Proof: The proof is by induction on $m = |Y|$. The base case when $m = 1$ is trivial. Now assume that the statement holds for all circuits with $|Y| < m$.

Let $0 = i_1 \leq i_2 \leq \ldots \leq i_m \leq \ell$ be the level indices of C as guaranteed by definition 11. Consider level $\ell' = i_{\lceil m/2 \rceil}$. Let $g_1, \ldots, g_w$ 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 $g_1, \ldots, g_w$ with new variables $Z_1, \ldots, Z_w$ and removing all gates below level ℓ'. Let $h_1, \ldots, h_w$ denote the output gates of A. (Note that these are just relabellings of $f_1, \ldots, f_w$.) Similarly, B is obtained from C by removing all nodes above layer ℓ' and making $g_1, \ldots, g_w$ 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 $Z = \{Z_1, \ldots, Z_w\}$: 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:

$$S = \left\{ R \subseteq Z \mid \begin{array}{l} \forall Z_i, Z_k \in R \text{ with } i \neq k, \text{ the sub-circuits} \\ \text{rooted at } g_i \text{ and } g_k \text{ are disjoint} \end{array} \right\}$$

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:

$$\begin{aligned} f_j &= h_j(g_1, \ldots, g_w); \qquad h_j = \sum_{\alpha \in S} c_{j,\alpha} Z^\alpha \\ \text{where } \; Z^\alpha &= \prod_{i=1}^{w} Z_i^{\alpha_i} \text{ and } \; c_{j,\alpha} \in \mathbb{K}[X, Y_A] \\ \text{Hence } \; f_j(X, Y) &= \sum_{\alpha \in S} c_{j,\alpha}(X, Y_A) g^\alpha(X, Y_B) \quad \text{where } g^\alpha(X, Y_B) = \prod_i g_i^{\alpha_i}(X, Y_B) \end{aligned}$$

Now using Observation 22 we have,

$$E_Y(f_j) = \sum_{\alpha \in S} E_Y(c_{j,\alpha} g^\alpha) = \sum_{\alpha \in S} E_{Y_A}(c_{j,\alpha}) E_{Y_B}(g^\alpha) \tag{2}$$

We need the following claim:

Claim 24 *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 of size $w.s_A$ and width w.*

Proof: The proof is by induction on the structure of the circuit. Let $\alpha = \alpha_1\alpha_2 \ldots \alpha_w$, where $\alpha_i \in \{0, 1\}$. The base case is when the sub-circuit rooted at g_j is a variable Z_i or $a \in \mathbb{K} \cup X \cup Y_A$. Then, $[c_{j,\alpha}]$ is set accordingly:

If $g_j = Z_i$ then

$$[c_{j,\alpha}] = \begin{cases} 1 \text{ for } \alpha_i = 1, \alpha_k = 0,\ i \neq k \\ 0 \text{ otherwise} \end{cases}$$

If $g_j = a \in \mathbb{K} \cup X \cup Y_A$ then,

$$[c_{j,\alpha}] = \begin{cases} a \text{ if } \alpha_i = 0, \forall i \\ 0 \text{ otherwise} \end{cases}$$

Induction step: case 1: $g_j = h_1 + h_2$, then $[c_{j,\alpha}] = [h_{1,\alpha}] + [h_{2,\alpha}]$.

case 2: $g_j = h_1 \times h_2$ then $[c_{j,\alpha}] = [h_{1,\alpha'}] + [h_{2,\alpha''}]$.

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, $[h_{1,\alpha}]$, $[h_{2,\alpha}]$, $[h_{1,\alpha'}]$ and $[h_{2,\alpha''}]$ are the corresponding coefficients available from inductive hypothesis.

The size of $[c_{j,\alpha}]$ thus obtained can blow up by factor of at most w and width remains unchanged. ■

Let $[c_{j,\alpha}]$ denote the circuit obtained in the above claim.

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 an addition sub-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 2, $E_Y(f_j)$ for each $1 \leq j \leq w$ can be computed by arithmetic circuits that compute all the $E_{Y_A}(c_{j,\alpha})$ and all the $E_{Y_B}(g^\alpha)$, and then put them together; such a circuit has size $T(w, m, s) = w \times |S| \times T(w, \lfloor m/2 \rfloor, ws_A) + |S| \times T(w, \lceil m/2 \rceil, s_B + w) + 2|S|$. As $|S| \leq 2^w$, we have,

$$\begin{aligned} T(w, m, s) &\leq w2^w T\left(w, \left\lfloor \frac{m}{2} \right\rfloor, ws_A\right) + 2^w T\left(w, \left\lceil \frac{m}{2} \right\rceil, s_B + w\right) + 2^{w+1} \\ s &= s_A + s_B \end{aligned}$$

By solving the recurrence we get the desired bound :
$T(w, m, s) = 2^{2(w+2)\log m} w^{2\log m} s + 2^{w+1} \log m = sm^{O(w)}$. ■

For VsSC circuits, the "upper half" circuit is not even multilinear. So we need to explicitly account for each monomial up to the overall degree, and compute the coefficient of each. We show that this is possible, if a quasi polynomial blow-up in size is allowed. Formally,

Lemma 25 *Let C be a layered arithmetic circuit size s on the variables $X \cup Y \cup Z$. Let d be the syntactic degree bound on C and w be its width. Let $f \in R[Z]$ be a polynomial computed by C, where $R = K[X, Y]$. Let $t = \langle t_1, \ldots, t_w \rangle$ be a degree sequence for variables from Z. Then $\mathsf{coeff}_f(Z^t)$ can be computed by a circuit of width $w + 2$ and size $O(s(d+1)^{2w})$, where $Z^t = \prod_{k=1}^{w} Z^{t_k}$.*

Proof: From the proof of Lemma 10, we have

$$\mathsf{coeff}_f(Z^t) = \sum_{i_1,\ldots,i_w \in \{0,\ldots,d\}} f(i_1, \ldots, i_w) \prod_{k=1}^{w} \mathsf{G}(x_k, i_k)$$

where

$$\mathsf{G}(x_k, i_k) = \frac{(-1)^{d-t_k} S_{d,t_k}(0, 1, \ldots, i_k - 1, i_k + 1, \ldots, d)}{i_k!(d - i_k)!(-1)^{(d-i_k)}}$$

The number of terms in the above sum is bounded by $(d + 1)^w$. We know that each S_{d,t_k} can be computed by a depth-3 arithmetic circuit of polynomial size $\mathsf{poly}(d)$; let $p(d)$ be the size upper bound on such a circuit. This circuit can easily be transformed into a width-3 circuit of the same size. Now, to compute $\mathsf{coeff}_f(Z^t)$, we compute each term in the above expression sequentially and accumulate the sum in an internal gate along the way. To compute a term, we need to evaluate f at a certain point $i_1 \ldots, i_w$ and then multiply it by $G(x_k, i_k)$. First we compute $f(i_1, \ldots, i_w)$ by using a copy of C, and then compute the product by serially computing the corresponding symmetric polynomials and multiplying by the inverse of the denominator (note that this value only depends on d and i_k's hence can be hardwired). Thus a term can be computed within a width of $\max\{w, 5\}$. To compute the overall sum, we need an extra gate to carry the partial sum. Thus the total width needed can be bounded by $\max\{w+1, 5\}$. The number of copies of C needed is bounded by $(d+1)^w$ and the total number of circuits for S_{d,t_k} is bounded by $(d + 1)^w w$. Hence the overall size can be bounded by $(d + 1)^w \times s + 2 \times (d + 1)^w \times w \times p(d)$. By [SW99], $p(d) = O(d^2)$; thus for $w \geq 2$ this is bounded by $O(s(d + 1)^{(2w)})$. ■

Proof:[of theorem 21] The first two statements follow directly from Lemma 23 For the third statement, let C be an arithmetic circuit of width w, size s and syntactic degree d. Now applying the strategy and using Lemma 25 for step 3, we can construct the required circuit C' through induction. Let $T(w, d, m)$ denote the size of the required circuit, then $T(w, d, m) \leq 2^w s(d + 1)^{2w} T(w, d, m/2)$. Solving the recurrence, it is easy to see that $T(w, d, m) = O(2^{w \log m} s^{\log m} (d + 1)^{2w \log m})$ which gives the required result. ■

6 Skew formula

In this section we consider the expressive power of exponential sums of polynomials computed by skew formula. Firstly, let us define exponential sum:

Definition 26 *Let $\mathcal{C}$ be an algebraic complexity class in Valiants' model. $\sum \cdot \mathcal{C}$ 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 $f_n(X) = \sum_{e\in\{0,1\}^{m'}} g_{m'+n}(X,e)$, where $m' \leq \mathsf{poly}(n)$. In this notation, $\mathsf{VNP} = \sum \cdot \mathsf{VP}$.*

It is well known that the complexity class NP is equivalent to $\exists \cdot \mathsf{P}$ and in fact even to $\exists \cdot \mathsf{F}$. A similar result holds in the case of Valiant's algebraic complexity classes too. Valiant has shown that $\mathsf{VNP} = \sum \cdot \mathsf{VF}$ (see [Bür00, BCS97]), and thus the polynomial g in the expression above 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 projection equivalent to the "permanent" polynomial, 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 ([Tod91]), 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 formula* of poly size, VSkewF? Recall that a circuit is said to be skew if every × (or ∧ 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 $\mathsf{VP} \subseteq \sum \cdot \mathsf{VP} = \sum \cdot \mathsf{VP}_e$, can we show that $\mathsf{VP}_{skew} \subseteq \sum \cdot \mathsf{VSkewF}$?

We show that this is not possible. We first give an equivalent characterization of VSkewF in terms of "sparse polynomials" (Lemma 27) placing it inside VAC^0, and then use it to show that $\sum \cdot \mathsf{VSkewF}$ is in fact contained in VSkewF (Theorem 29).

6.1 A characterization of VSkewF

Lemma 27 *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.*
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 $m_1, \ldots, m_t$, 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 $O(td)$. ■

Corollary 28 $\mathsf{VSAC}^0 \subset \mathsf{VSkewF} \subset \mathsf{VAC}^0$.

Proof: The containments follow directly from Lemma 27. To see why they are proper: (1) Even over the Boolean setting, the function $\oplus_{i=1}^{\log n} x_i$ is in SkewF but not in SAC^0. Any Boolean function sensitive to only $O(\log n)$ of its n inputs is in SkewF. Functions computed by a VSAC^0 circuit have $O(1)$ degree, and so cannot equal the class of poly-degree poly-support polynomials VSkewF. (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. ■

6.2 An upper bound for $\sum.\mathsf{VSkewF}$

Theorem 29 *Let $f \in \mathbb{K}[X]$ be expressible as $f(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where ϕ has a poly size* skew *formula and $m \leq \mathsf{poly}(n)$. Then $f \in \mathsf{VSkewF}$.*
In other words, $\sum \cdot \mathsf{VSkewF} \subseteq \mathsf{VSkewF}$.

Proof: Since $\phi(X, Y)$ (where $X = X_1, \ldots, X_n$ and $Y = Y_1, \ldots, Y_m$) has a poly size skew formula, by Lemma 27 we know that the number of non-zero monomials in ϕ is bounded by some polynomial $q(n, m)$. Hence the number of non-zero monomials in $\phi(X, Y)|_X$ (*i.e.* , monomials in X with coefficients from $\mathbb{Z}[Y]$) and hence in $f(X)$, is also bounded by $q(n, m)$.

For any $\alpha \in \mathbb{N}^n$, consider the monomial $X^\alpha = \prod_{\alpha_i} X_i^{\alpha_i}$, and define the set S_α as

$$S_\alpha = \{\beta \in \{0,1\}^m \mid X^\alpha Y^\beta \text{ has a non-zero coefficient } a_{\alpha,\beta} \text{ in } \phi\}$$

Clearly, for each α, we have $|S_\alpha| \leq q(n, m)$.

Since $\phi(X, Y)$ is evaluated only at Boolean settings of Y, we can assume, without loss of generality, that it is multilinear in Y. So it can be written as

$$\phi(X, Y) = \sum_{\alpha \in \mathbb{N}^n} \sum_{\beta \in \{0,1\}^m} a_{\alpha,\beta} X^\alpha Y^\beta$$

Hence we have the following:

$$\begin{aligned} f(X) &= \sum_{e \in \{0,1\}^m} \sum_{\alpha \in \mathbb{N}^n} \sum_{\beta \in \{0,1\}^m} a_{\alpha,\beta} X^\alpha e^\beta \\ &= \sum_{\alpha \in \mathbb{N}^n} \left(X^\alpha \sum_{\beta \in S_\alpha} \left[a_{\alpha,\beta} \sum_{e \in \{0,1\}^m} e^\beta \right] \right) \end{aligned}$$

Therefore,

$$f(X) \;=\; \sum_{\alpha \in \mathbb{N}^n} \left(X^\alpha \sum_{\beta \in S_\alpha} a_{\alpha,\beta} 2^{m-l_\beta} \right)$$

where l_β = number of 1's in the bit vector $\beta \in \{0,1\}^m$.

Then the coefficient c_α of X^α in $f(X)$ is given by $\sum_{\beta \in S_\alpha} a_{\alpha,\beta} 2^{m-l_\beta}$. Now by hardwiring these coefficients along with X^αs (note that there are only polynomially many such αs) it is easy to see that $f(X)$ can be computed by a skew formula. ■

Thus, it is not possible to express the determinant polynomial in $\sum$.VSkewF since it has exponentially many monomials.

6.3 Multilinear Versions

Here we consider the multilinear versions of the skew formula. From Lemma 27, we know that VSkewF is characterized by polynomials with polynomial many coefficients. The construction yields, for any multilinear polynomial computed by a skew formula, an equivalent skew formula which is syntactic multilinear. Hence the notion of multilinearity and syntactic multilinearity are the same for skew formula.

Since any multilinear polynomial that can be computed by a VSAC^0 circuit has a small number of monomials, the containments and separations of corollary 28 hold in the syntactic multilinear case too. Also, note that the polynomial $\prod_i (x_i + y_i)$ is multilinear, and can be computed by a sm-AC^0 circuit.

Corollary 30 sm-SAC^0 $\subset$ sm-VSkewF $\subset$ sm-AC^0

Conclusion and Open questions

We proposed a notion of "small space" for algebraic computations in terms of the circuit width. VL was defined as class of polynomials computed by log width circuits with certain degree and constraints on coefficients. However it is easy to see that our definition of VWIDTH$(S(n))$ can be extended to polynomials with arbitrary coefficients from $\mathbb{K}$. Only Theorem 7 does not work under this definition as VPSPACE contains only polynomials with integer coefficients.

Having a reasonable upper bound for VL seems to be a hard task: as VBP can be seen as a natural arithmetic version of NL, we would like to have VL contained inside VBP.

Later on we introduced the notion of read-once certificates and read-once exponential sums of arithmetic circuits. It is shown that with this definition, the 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. However, in the case of VsSC^i we could prove only an upper bound of VQP, *i.e.* $\Sigma^R \cdot VsSC^i \subseteq$ VQP (Theorem 21). For the case of $\Sigma^R \cdot \mathsf{VL}$ the best upper bound one could give is only VNP which is obvious from definition itself.

Are VNC^1 and VsSC^0 separate? The study of read once exponential sums throws in this doubt: Is VsSC^0 really more powerful than VNC^1? Since we don't have a nice definition of read once certificates for depth bounded circuits, we use the equivalence $\mathsf{VBWBP} = \mathsf{VNC}^1$ for this purpose. From Theorem 18, we have $\Sigma^R \cdot \mathsf{VBWBP} = \mathsf{VBWBP}$, hence we can say that $\Sigma^R \cdot \mathsf{VNC}^1 = \mathsf{VNC}^1$. On the other hand the best known upper bound for $\Sigma^R \cdot \mathsf{VsSC}^0$ is VQP. Thus on the one hand showing $\mathsf{VNC}^1 = \mathsf{VsSC}^0$ will bring $\Sigma^R \cdot \mathsf{VsSC}^0$ all the way down to VNC^1 and showing a super-polynomial formula size lower bound for VsSC^0 could separate VsSC^0 from VNC^1. However the second one is going to be much harder.

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 $\Sigma^R \cdot \mathsf{VL} \subseteq \mathsf{VP}$? Even in the case of VSC^0, it will be interesting to see an upper bound of VP, i.e, is $\Sigma^R \cdot \mathsf{VSC}^0 \subseteq \mathsf{VP}$?
- Is there any natural family of polynomials complete for VL?

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 2009.

[Bar89] David.A.Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BCS97] P Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory.* Springer-Verlag, 1997.

[BCSS97] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation.* Springer, 1997.

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory.* Algorithms and Computation in Mathematics. Springer-Verlag, 2000.

[CDL01] A Chiu, G Davida, and B Litow. Division in logspace-uniform NC^1. *RAIRO Theoretical Informatics and Applications*, 35:259–276, 2001.

[CMTV98] Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.

[dN06] Paulin Jacobé de Naurois. A Measure of Space for Computing over the Reals. In *CiE*, pages 231–240, 2006.

[Jan08] Maurice J. Jansen. Lower bounds for syntactically multilinear algebraic branching programs. In *MFCS*, pages 407–418, 2008.

[JR09] Maurice J. Jansen and B. V. Raghavendra Rao. Simulation of arithmetical circuits by branching programs with preservation of constant width and syntactic multilinearity. In *CSR*, pages 179–190, 2009.

[KP07a] Pascal Koiran and Sylvain Perifel. VPSPACE and a Transfer Theorem over the Complex Field. In *MFCS*, pages 359–370, 2007.

[KP07b] Pascal Koiran and Sylvain Perifel. VPSPACE and a Transfer Theorem over the Reals. In Wolfgang Thomas and Pascal Weil, editors, *STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 417–428. Springer, 2007.

[LMR07] Nutan Limaye, Meena Mahajan, and B V Raghavendra Rao. Arithmetizing classes around NC^1 and l. In Patrice Enjalbert, Alain Finkel, and Klaus W. Wagner, editors, *STACS*, volume 665 of *Lecture Notes in Computer Science*, pages 477–488. Springer, 2007.

[Mal07] Guillaume Malod. The complexity of polynomials and their coefficient functions. In *IEEE Conference on Computational Complexity*, pages 193–204, 2007.

[Mic89] Christian Michaux. 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, 1989.

[MP06] Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. In *MFCS*, pages 704–716, 2006.

[NW95] Noam Nisan and Avi Wigderson. On the complexity of bilinear forms. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 723–732, New York, NY, USA, 1995. ACM.

[SW99] Amir Shpilka and Avi Wigderson. Depth-3 arithmetic formulae over fields of characteristic zero. In *IEEE Conference on Computational Complexity*, pages 87–, 1999.

[Tod91] Seinosuke Toda. Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. Comp. Sci. and Inf. Math., Univ. of Electro-Communications, Tokyo, 1991.

[Tza08] Iddo Tzamaret. *Studies in Algebraic and Propositional Proof Complexity.* PhD thesis, Tel Aviv University, 2008.

[Val76] Leslie G. Valiant. Graph-theoretic properties in computational complexity. *Journal of Computer and System Sciences*, 13:278–285, 1976.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Ven92] H. Venkateswaran. Circuit definitions of nondeterministic complexity classes. *SIAM Journal on Computing*, 21:655–670, 1992.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Springer-Verlag New York Inc., 1999.

7 Appendix

7.1 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 [BCSS97]. The BSS model is defined for computation over the field $\mathbb{R}$ of real numbers. We present the version used in [dN06].

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 $\mathcal{A} = \{A_1, \ldots, A_k\}$, 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 $\phi_M : \mathbb{R}^* \to \mathbb{R}$ 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 a the number of work tape cells used by the machine as the space used. Michaux ([Mic89]) showed the following:

Proposition 31 ([Mic89]) *Let $L \subseteq \mathbb{R}$ be a language computed by a machine M in time t. Then there is machine M' and a constant k such that M' computes L in unit space k.*