Electronic Colloquium on Computational Complexity, Report No. 87 (2017)

On Weak-Space Complexity over Complex Numbers

Pushkar S Joglekar[1], B. V. Raghavendra Rao[2], and Siddhartha Sivakumar[2]

[1] Vishwakarma Institute of Technology, Pune, India joglekar.pushkar@gmail.com
[2] Indian Institute of Technology Madras, Chennai, India, bvrr@cse.iitm.ac.in, sith1992@gmail.com

Abstract. Defining a feasible notion of space over the Blum-Shub-Smale (BSS) model of algebraic computation is a long standing open problem. In an attempt to define a right notion of space complexity for the BSS model, Naurois [CiE, 2007] introduced the notion of weak-space. We investigate the weak-space bounded computations and their plausible relationship with the classical space bounded computations. For weak-space bounded, division-free computations over BSS machines over complex numbers with $\stackrel{?}{=} 0$ tests, we show the following:

1. The Boolean part of the weak log-space class is contained in deterministic log-space, i.e.,

$$\mathsf{BP}(\mathsf{LOGSPACE_W}) \subseteq \mathsf{DLOG}.$$

2. There is a set $L \in \mathsf{NC}^1_{\mathbb{C}}$ that cannot be decided by any deterministic BSS machine whose weak-space is bounded above by a polynomial in the input length, i.e., $\mathsf{NC}^1_{\mathbb{C}} \not\subseteq \mathsf{PSPACE_W}$.

The second result above resolves the first part of Conjecture 1 stated in [6] over complex numbers and exhibits a limitation of weak-space. The proof is based on the structural properties of the semi-algebraic sets contained in $\mathsf{PSPACE_W}$ and the result that any polynomial divisible by a degree-$\omega(1)$ elementary symmetric polynomial cannot be sparse. The lower bound on the sparsity is proved via an argument involving Newton polytopes of polynomials and bounds on number of vertices of these polytopes, which might be of an independent interest.

1 Introduction

The theory of algebraic computation aims at classifying algebraic computational problems in terms of their intrinsic algebraic complexity. Valiant [25] developed a non-uniform notion of complexity for polynomial evaluations based on arithmetic circuits as a model of computation. Valiant's work lead to intensive research efforts towards classifying polynomials based on their complexity. (See [23] for a survey). Valiant's model is non-uniform and it does not allow comparison operation on the values computed. This lead to the seminal work by Blum, Shub and Smale [3] where a real and complex number counterpart of Turning machines, now known as BSS machines has been proposed.

Blum, Shub, Smale and Cucker [2] defined the complexity classes such as $\mathsf{P}_{\mathbb{R}}$ and $\mathsf{NP}_{\mathbb{R}}$ in analogy to the classical complexity classes P and NP and proposed the conjecture: $\mathsf{P}_{\mathbb{R}} \neq \mathsf{NP}_{\mathbb{R}}$. Several natural problems such as Hilbert's Nullstellensatz, Feasibility of quadratic equations are complete for the class $\mathsf{NP}_{\mathbb{R}}$ [2]. Further, there has been a significant amount of work on the structural aspects of real computation with various restrictions placed on the computational model. See [17] for a survey of these results.

One of the fundamental objectives of algebraic complexity theory is to obtain transfer theorems, i.e., to translate separations of algebraic complexity classes to either the Boolean world or other models of algebraic computation. Though establishing a relation between the BSS model of computation and the classical Turing machine is a hard task, Fournier

ISSN 1433-8092

and Koiran [7] showed that proving super polynomial time lower bounds against the BSS model would imply separation of classical complexity classes. Also, there has been a study of *algebraic* circuits leading to the definition of parallel complexity classes $\mathsf{NC}_{\mathbb{R}}$. In contrast to the Boolean counterparts, Cucker [4] showed that there are sets in $\mathsf{P}_{\mathbb{R}}$ that cannot have efficient parallel algorithms, i.e., $\mathsf{P}_{\mathbb{R}} \neq \mathsf{NC}_{\mathbb{R}}$.

One of the pre-requisites for transfer theorems would be a comparison with the complexity classes in the Boolean world. One approach towards this is restricting the BSS machines over Boolean inputs. A restriction of a real complexity class to Boolean inputs is called Boolean part and is denoted using the prefix the prefix BP, e.g, $\mathsf{BP}(\mathsf{P}_{\mathbb{R}})$ denotes the class of all languages over $\{0,1\}^*$ that can be decided by polynomial time bounded BSS machines [2, 10]. Koiran [10] did an extensive study of Boolean parts of real complexity classes. Cucker and Grigoriev [5] showed that $\mathsf{BP}(\mathsf{P}_{\mathbb{R}}) \subseteq \mathsf{PSPACE/poly}$. Further, Allender et.al, [1] studied computational tasks arising from numerical computation and showed that the task of testing positivity of an integer represented as an arithmetic circuit is complete for the class $\mathsf{BP}(\mathsf{P}_{\mathbb{R}})$.

Though the notion of time complexity has been well understood in the real model of computation, it turned out that, setting up a notion of space is difficult. Michaux [18] showed that any computation over the real numbers in the BSS model can be done with only a constant number of cells. This rules out the possibility of using the number of cells used in the computation as a measure of space. Despite the fact that there has been study of parallel complexity classes, a natural measure of space that leads to interesting space complexity classes in analogy with the classical world is still missing.

Naurois [6] proposed the notion of weak-space for computation over real numbers in the BSS model. This is motivated by the weak BSS model of computation proposed by Koiran [12]. The notion of weak-space takes into account the number of bits needed to represent the polynomials representing each cell of a configuration. (See Section 2 or [6] for a formal definition.) Based on this notion of space Naruois [6] introduced weak-space classes $\mathsf{LOGSPACE}_{\mathsf{W}}$ and $\mathsf{PSPACE}_{\mathsf{W}}$ as analogues of the classical space complexity classes DLOG and PSPACE and showed that $\mathsf{LOGSPACE}_{\mathsf{W}}$ is contained in $\mathsf{P}_W \cap NC^2_{\mathbb{R}}$, where P_W is the class of sets decidable in weak polynomial time [12]. The notion of weak-space enables space bounded computations to have a finite number of configurations, and hence opening the scope for possible analogy with the classical counterparts. However, [6] left several intriguing questions open. Among them; a real analogue of NC^1 versus DLOG, and an upper bound for the Boolean parts of weak space classes.

In this paper, we continue the study of weak-space classes initiated by Naurois [6] and investigate weak-space bounded division free computations where equality is the only test operation allowed. In particular, we address some of the questions left open in [6].

Our Results: We begin with the study of Boolean parts of weak space complexity classes. We show that the Boolean part of $\mathsf{LOGSPACE}_{\mathsf{W}}$ is contained in DLOG. (See Theorem 2.) Our proof involves a careful adaptation of the constant elimination technique used by Koiran [11] to weak space bounded computation.

We show that there is a set $L \in \mathsf{NC}^1_{\mathbb{F}}$ that cannot be accepted by any polynomial weak-space bounded BSS machine, i.e., $\mathsf{NC}^1_{\mathbb{F}} \not\subset \mathsf{PSPACE}_{\mathsf{W}}$ (Theorem 3 and Corollary 1) where

$\mathbb{F} \in \{\mathbb{C}, \mathbb{R}\}$. This resolves the first part of the Conjecture 1 in [6] where the computation is division free and only equality tests are allowed. Also, this result is in stark contrast to the Boolean case, where $\mathsf{NC}^1 \subseteq \mathsf{DLOG}$.

Our Techniques: For the proof of Theorem 3, we consider the restriction $L_n = L \cap \mathbb{F}^n$ for a set $L \in \mathsf{LOGSPACE_W}$ and obtain a characterization for the defining polynomials of L_n as a semi-algebraic set in $\mathbb{F}^n$. Then using properties of the Zarisky topology, we observe that if L_n is an irreducible algebraic set, then the defining polynomial for L_n has small weak size. With this, it suffices to obtain a set $L \in \mathsf{NC}^1_{\mathbb{F}}$ such that each slice L_n is a hyper-surface such that any non-trivial hyper-surface containing it cannot have sparse polynomial as its defining equations. We achieve this by considering the elementary symmetric polynomial of degree $n/2$ as the defining equation for L_n. For every polynomial multiple of the elementary symmetric polynomial, we prove a lower bound on its sparsity by appealing to the structure of Newton polytopes of these polynomials. (See Theorem 4 for a precise statement.)

Related Results Koiran and Perfiel [14, 13] have studied the notion of polynomial space in Valiant's algebraic model and obtained transfer theorems over the real and complex numbers. Mahajan and Rao [16] obtained small space complexity classes in Valiant's algebraic model. To the best of our knowledge, apart from these, and the results by Michaux [18] and Naurois [6], there have been no significant study of space complexity classes in the broad area of algebraic complexity theory.

Organization of the paper In Section 2, we briefly review the BSS model of computation, and provide all necessary but non-standard definitions used in the paper. In Section 3 we look at the Boolean part of $\mathsf{LOGSPACE_W}$. Section 4 we prove the main theorem (Theorem 3) of the paper. Section 5 proves Corollary 2 which is an important component in the proof of Theorem 3.

2 Preliminaries

An overview of the BSS model of real computation

We give a brief description of a Blum-Shub-Smale (BSS) machine over $\mathbb{F}$. For details, the reader is referred to [3].

Definition 1. *A Blum-Shub-Smale (BSS) machine M over $\mathbb{F}$ with parameters $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$ with $k \geq 0$ and an admissible input $Y \subseteq \mathbb{F}^\infty$ is a Random Access Machine with a countable number of* registers *(or* cells*) each capable of a storing a value from $\mathbb{F}$. The machine is permitted to perform three kinds of operations:*

Computation: Perform $c_l = c_i$ op c_j, where c_i, c_j and c_l are either cells of M or among the parameters and $op \in \{+, \times, -\}$ and move to the next state.

Branch (test): Perform the test $c \stackrel{?}{=} 0$ for some cell c and move to the next state depending on the result, i.e., branch as per the outcome of the test.

Copy: $c_i = c_j$, copy the value of the cell c_j into c_i. Here c_j can also be one of the parameters $\alpha_1 \ldots, \alpha_k$ of M.

It should be noted that in the definition of a real BSS machine the test instruction is usually $\overset{?}{\geq} 0$ rather than equality. Throughout the paper, we restrict ourselves to BSS machines where the test operation is $\overset{?}{=} 0$. Also, in general, BSS machines allow the division operation, however, we restrict to BSS machines where division is not allowed.

Notion of acceptance and rejection of an input, configurations and time complexity of computation can be defined similar to the case of classical Turing Machines, see [2] for details.

For a BSS machine that halts on all admissible inputs, the set accepted by M is denoted by $L(M)$. For an input $x \in \mathbb{F}^n$, the size of the input x is n.

Definition 2 (Complexity Class $\mathsf{P}_{\mathbb{F}}$). *[2] Let $\mathbb{F}$ be a field of real or complex numbers then the complexity class $\mathsf{P}_{\mathbb{F}}$ is defined as the set of all languages $L \subseteq \mathbb{F}^{\infty}$ such that, there is a polynomial time BSS machine accepting L.*

The classes $\mathsf{NP}_{\mathbb{F}}$ and $\mathsf{EXP}_{\mathbb{F}}$ are defined analogously.

A BSS machine M is said to be *constant free* if the number of parameters $k = 0$. The restrictions of the above defined classes that are based on constant-free machine are denoted with a superscript of $\mathsf{0}$, e.g., $\mathsf{P}^{\mathsf{0}}_{\mathbb{F}}$ denotes the class of all sets in $\mathsf{P}_{\mathbb{F}}$ that are accepted by polynomial time bounded constant-free BSS machines.

Arithmetic and Algebraic circuits An *arithmetic circuit* is an implicit representation of a polynomial. It is a labelled directed acyclic graph where vertices have in-degree either zero or two. Vertices of zero in-degree are called *input* gates and are labelled by elements in $\mathbb{F} \cup \{x_1, \ldots, x_n\}$. Vertices of in-degree two are called *internal* gates and have their labels from $\{\times, +\}$, and vertices of zero out-degree are called *output* gates. Every gate of an arithmetic circuit computes a polynomial over $\mathbb{F}$. The polynomial(s) computed by an arithmetic circuit is the (set of) polynomial(s) computed at its output gate(s). Size of an arithmetic circuit is the number of gates in it. Depth of an arithmetic circuit is the longest length of a path from an input gate to an output gate in the circuit.

An *algebraic circuit* is an arithmetic circuit where in addition to the $\times$ and $+$ gates a test gate $\overset{?}{=} 0$ is allowed. A test gate has a single input and outputs either 0 or 1 depending on the outcome of the test. Size and depth of algebraic circuits are defined analogously. For the purpose comparing with BSS complexity classes, we assume that algebraic circuits have a single output gate which is a $\overset{?}{=} 0$ gate. The following complexity classes are defined based on algebraic circuits.

Definition 3. *[2] Let $\mathbb{F}$ be a field of real or complex numbers then the complexity class $\mathsf{NC}^i_{\mathbb{F}}$ is defined as, the set of all languages $L \subseteq \mathbb{F}^*$, for which there is an algebraic circuit family $(C_n)_{n \geq 0}$, size of C_n is polynomial in n and depth of C_n is $O((\log n)^i)$ such that for all $n \geq 0$ and $x \in \mathbb{F}^n$, x is in L iff $C_n(x) = 1$.*

Definition 4. *[17] Let $\mathbb{F}$ be a field of real or complex numbers then the complexity class $\mathsf{PAR}_{\mathbb{F}}$ is defined as, the set of all languages $L \subseteq \mathbb{F}^*$, for which there is an algebraic circuit family $(C_n)_{n \geq 0}$ such that depth of C_n is $n^{O(1)}$ and for all $n \geq 0$ and $x \in \mathbb{F}^n$, x is in L iff $C_n(x) = 1$.*

Note that in the definition of $\mathsf{PAR}_{\mathbb{F}}$ above, size of the circuit is allowed to be exponential. Further, we have assumed that the admissible input is $\mathbb{F}^*$ in our definitions, though, in general, an algebraic circuit need to output the correct decision only on admissible inputs. Further, for comparison with BSS machine based classes, a suitable notion of uniformity is required. For more details see [2].

Weak Time In order to be able to compare BSS complexity classes with classical Boolean complexity classes, Koiran [12] introduced a weak notion of time in the BSS model, called the *weak BSS model*. Intuitively, the weak BSS model takes the size of the integers being represented in the cells of the BSS machine during the process of computation. In the weak BSS model, an arithmetic operation $x = y \ \mathsf{op}\ z$ comes associated with a cost. Cost of an operation is the maximum degree of the resulting polynomial representing x and maximum of bit sizes of coefficients. (See [12] for more details.) Using this notion of cost, [12] defined the weak variants of complexity classes. The corresponding classes are denoted with a subscript w. See [12] for a detailed exposition.

Weak Space Following the notion of weak time defined by Koiran [12], Naurois [6], introduced the notion of weak space for BSS machines. To begin with, we need a measure of weak size of polynomials with integer coefficients. Let $g \in \mathbb{Z}[x_1, \ldots, x_n]$ be a polynomial of degree d. The binary encoding $\phi(m)$ corresponding to a monomial $m = x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_k}^{\alpha_k}$ is simply concatenation of $\lceil \log n \rceil$ bit binary encoding of index i_j and $\lceil \log d \rceil$ bit binary encoding of exponent α_j for $j \in [k]$, *i.e.*,

$$\phi(m) = \langle i_1 \rangle \langle \alpha_1 \rangle \cdot \langle i_2 \rangle \langle \alpha_2 \rangle \ldots \langle i_k \rangle \langle \alpha_k \rangle$$

where $\langle i_j \rangle, \langle \alpha_j \rangle$ denotes binary encoding of integers i_j and α_j respectively. Let $g = \sum_{m \in M} g_m m$ where $g_m \neq 0$ is the coefficient of monomial m in g and $M = \{m_1, m_2, \ldots, m_s\}$ be the set of monomials of g with non-zero coefficients. Then the binary encoding of g is

$$\phi(g) = b_1 \langle g_{m_1} \rangle \phi(m_1) \cdot b_2 \langle g_{m_2} \rangle \phi(m_2) \cdot \cdots \ldots b_s \langle g_{m_s} \rangle \phi(m_s)$$

where $b_i = 1$ if $g_{m_i} \geq 0$ else $b_i = 0$ and $\langle g_{m_i} \rangle$ denotes $\lceil \log C \rceil$-bit binary encoding of g_{m_i} for $i \in [s]$ where $C = max_i |g_{m_i}|$. We denote length of encoding $\phi(g)$ by $S_{weak}(g)$ and call it weak size of polynomial g. It is easy to see that $S_{weak}(g) \leq s(n(\lceil \log n \rceil + \lceil \log d \rceil) + 1 + \lceil \log C \rceil)$.

Remark 1. Our definition above, there is a possibility that bit size $\phi(g)$ depends on the labelling of the variables. For example, if $g = x_{n-1} x_n$, when $\phi(g)$ would require $2 \log n + 1$ bits, whereas, if we index x_{n-1} by number 1 and x_n by 2, then $\phi(g)$ requires ar most 6 bits. Due to this, Nauraois [6] in his definition allowed a cyclic shift of the indices of variables in the binary encoding. It should be noted that this is useful only when the polynomial g depends on a small set of variables. However, when we need the computation to depend on all of the variables, the definition above is without loss of generality.

Definition 5. (Weak-space complexity) *Let M be a BSS machine with parameters $\alpha_1, \alpha_2 .., \alpha_m \in \mathbb{F}$, and an input $x = (x_1, x_2, \ldots, x_n)$. Let $\mathcal{C}_M(x)$ denote the set of all configurations of M on x*

reachable from the initial configuration. For a configuration $c \in \mathcal{C}_M(x)$*, let* $f_1^{(c)}, f_2^{(c)}, \ldots, f_r^{(c)}$ *be the polynomial functions representing the non-empty cells in the configuration such that*

$$f_i^{(c)}(x_1, x_2, \ldots, x_n) = g_i^{(c)}(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_m)$$

where $g_i^{(c)} \in \mathbb{Z}[x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m]$ *for* $i \in [m]$*. Define the weak size of a configuration* c *as* $S_{\mathsf{weak}}(c) = \sum_{j=1}^{r} \mathsf{S}_{\mathsf{weak}}(g_j^{(c)})$ *Then the weak-space complexity of* M *is defined as*

$$WSpace_M(n) = \max_{x \in \mathbb{F}^n} \max_{c \in \mathcal{C}_M(x)} \mathsf{S}_{\mathsf{weak}}(c).$$

We say that a BSS machine M is said do be s weak-space bounded, if $WSpace_M(n) \leq s(n)$. The following concrete weak space classes have been defined in [6].

Definition 6 (Complexity class $\mathsf{SPACE_W}(s)$). *For a non-decreasing space constructible function* s*,* $\mathsf{SPACE_W}(s)$ *is the set of all languages* $L \subseteq \mathbb{F}^*$*, for which there is a BSS machine* M *over* $\mathbb{F}$ *such that* $L(M) = L$ *and* $WSpace_M(n) = O(s(n))$*.*

Note that we have omitted the subscript $\mathbb{F}$ in the above definition, this is not an issue since the field will always be clear from the context. The following inclusions are known from [6].

Proposition 1 *[6]* $\mathsf{LOGSPACE_W} \subseteq \mathsf{P_w} \cap \mathsf{NC}^2_{\mathbb{R}}$; *and* $\mathsf{PSPACE_W} \subset \mathsf{PAR}_{\mathbb{R}}$.

For definition of an algebraic variety and the Zariski topology, the reader is referred to [22]. The elementary symmetric polynomial of degree d is defined as:

$$\mathsf{sym}_{n,d}(x_1, \ldots, x_n) = \sum_{S \subseteq [n], |S| = d} \prod_{i \in S} x_i,$$

where $[n] = \{1, \ldots, n\}$.

Convex Polytopes For the proof of our lowebound result in Section 5 we need to review some basic concepts about convex polytopes. For a detailed exposition on convex polytopes, see e.g. [9], [26].

A point set $K \subseteq \mathbb{R}^d$ is *convex* if for any two points $x, y \in K$, the point $\lambda x + (1 - \lambda)y$ is in K for any λ, $0 \leq \lambda \leq 1$. The intersection of convex sets is convex. For any $K \subseteq \mathbb{R}^d$, the intersection of all convex sets containing K is called as *convex-hull* of K, $conv(K) = \bigcap\{T \subseteq \mathbb{R}^d | K \subseteq T, T \text{ is convex}\}$.

From the above definition and a simple inductive argument it follows that

Lemma 1. *If* $K \subseteq \mathbb{R}^d$ *and* $x_1, x_2, \ldots, x_n \in K$ *then* $\sum_{i=1}^n \lambda_i x_i \in conv(K)$ *where* $\lambda_i \geq 0$ *and* $\sum_{i=1}^n \lambda_i = 1$ *and if* $K = \{x_1, \ldots, x_n\}$ *is a finite set of points then* $conv(K) = \{\sum_{i=1}^n \lambda_i x_i | \lambda_i \geq 0$ *and* $\sum_{i=1}^n \lambda_i = 1\}$*.*

Definition 7. *(Convex Polytope) A convex-hull of a finite set of points in* $\mathbb{R}^d$ *is called as convex polytope.*

Let $P = conv(\{x_1, \ldots, x_n\}) \subset \mathbb{R}^d$ be a convex polytope. Then the dimension of P (denoted as $dim(P)$) is the dimension of the affine space $\{\sum_i \lambda_i x_i | \lambda_i \in \mathbb{R}, \sum_i \lambda_i = 1\}$. Clearly if $P \subset \mathbb{R}^d$ then $dim(P) \leq d$.

We can equivalently think of convex polytopes as bounded sets which are intersections of finitely many closed half spaces in some $\mathbb{R}^d$. More precisely,

Theorem 1. *(Chapter 1, [26]) P is convex-hull of finite set of points in $\mathbb{R}^d$ iff there exists $A \in \mathbb{R}^{m \times d}$ and $z \in \mathbb{R}^m$ such that the set $\{x \in \mathbb{R}^d | Ax \leq z\}$ is bounded and $P = \{x \in \mathbb{R}^d | Ax \leq z\}$.*

Definition 8. *(Face of Polytope) Let P is a convex polytope in $\mathbb{R}^d$. For $a = (a_1, a_2, \ldots, a_d) \in \mathbb{R}^d$ and $b \in \mathbb{R}$ we say the linear inequality $\langle a, x \rangle \leq b$ (where $\langle a, x \rangle = \sum_{i=1}^d a_i x_i$) is valid for P if every point $x = (x_1, \ldots, x_d) \in P$ satisfy it.* A face of *P is any set of points in $\mathbb{R}^d$ of the form $P \cap \{x \in \mathbb{R}^d | \langle a, x \rangle = b\}$ for some $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $\langle a, x \rangle \leq b$ is a valid linear inequality for P.*

From the above definition and theorem 1 it is clear that every face of a convex polytope is also a convex polytope. So we can use notion of dimension of convex polytope to talk about dimension of a face of a convex polytope. The faces of dimension 0 are called as the *vertices* of the polytope. Following proposition gives useful criteria for a point $v \in P$ to be a vertex of P. For the proof of following standard propositions refer to Chapter 1,2 of [26].

Proposition 2 *For a convex polytope P, a point $v \in P$ is vertex of P iff for any $n \geq 1$, and any $x_1, \ldots, x_n \in P$, $v \neq \sum_{i=1}^n \lambda_i x_i$ for $0 \leq \lambda_i < 1, \sum_i \lambda_i = 1$*

Proposition 3 *Every convex polytope P is convex-hull of set of its vertices, $P = conv(ver(P))$ and if $P = conv(S)$ for finite S then $ver(P) \subseteq S$, where $ver(P)$ denotes the set of vertices of a polytope P.*

3 Boolean parts of weak space classes

Though the BSS model is intended to capture the intrinsic complexity of computations over real and complex numbers, it is natural to study the power of such computations restricted to the Boolean input. The Boolean parts of real/complex complexity classes have been well studied in the literature [1]. We consider Boolean parts of the weak- space classes introduced by Naurois [6]

Definition 9. *Let C be a complexity class in the BSS model of computation, then the Boolean part of C denoted by $\mathsf{BP}(C)$ is the set $\mathsf{BP}(C) = \{L \cap \{0,1\}^* \mid L \in C\}$*

We observe that the Boolean part of $\mathsf{LOGSPACE_W}$ is contained in DLOG, i.e. the class of languages accepted deterministic logarithmic space bounded Turing Machines.

Theorem 2. *For $\mathbb{F} \in \{\mathbb{C}, \mathbb{R}\}$, $\mathsf{BP}(\mathsf{LOGSPACE_W}) \subseteq \mathsf{DLOG}$.*

Proof. Let $L \in \mathsf{LOGSPACE_W}$ and M be a BSS machine over $\mathbb{F}$ with $WSpace_M(n) = s(n) = c \log n$ for some $c > 0$ and such that $\forall x \in \mathbb{F}^*,\ x \in L \iff M$ accepts x. Our proof is a careful analysis of the constant elimination procedure developed by Koiran [11]. The argument is divided into three cases:
Case 1: Suppose that M does not use any constants from $\mathbb{F}$. Let $x_1, \ldots, x_n \in \{0,1\}$ be an input. Construct a Turing Machine M' that on input $x_1, \ldots, x_n \in \{0,1\}$ simulates M as follows. M' stores content of each cell of M explicitly as a polynomial. For each step of M:

1. If the step is an arithmetic operation, then M' explicitly computes the resulting polynomial and stores it in the target cell and proceeds.
2. If the step is a comparison operation, then M' evaluates the corresponding polynomial corresponding and proceeds to the next step of M.

Since the total number of bits required to store all of the polynomials in any given configuration is bounded by $c \log n$ and the arithmetic operations on log-bit representable polynomials can be done in deterministic log-space, it is not difficult to see that the resulting Turing Machine M is log-space bounded.
Case 2: M uses algebraic constants. Suppose $\beta_1, \ldots, \beta_k \in \mathbb{F}$ are the algebraic constants used in M. We begin with the special case when $k = 1$. Let $p_1(x)$ be the minimal polynomial of β_1 with coefficients in $\mathbb{Z}$. Let d be the degree of p_1. We show that Koiran's [11] technique for elimination of algebraic constants can indeed be implemented in weak log-space. We view the content of each cell of M on a given input $x_1, \ldots, x_n \in \{0,1\}$ as a polynomial in $x_1, \ldots, x_n$ and a new variable y_1. For any polynomial $q(x_1, \ldots, x_n, y_1)$ with integer coefficients, $q(x_1, \ldots, x_n, \beta_1) = 0$ if and only if $q(x_1, \ldots, x_n, y_1) = 0 \mod p_1$. Consider the Turing machine M' that simulates M as follows. M' stores contents of each cell of M as polynomial $p(x_1, \ldots, x_n, y_1) \mod p_1$. Note that every such polynomial has degree d in the variable y_1. For each step of the machine M, the new Turing machine M' does the following:

1. If the step is an arithmetic (add or multiply) operation, then perform the same arithmetic operation on the corresponding polynomials modulo p_1 and store the resulting polynomial in the polynomial corresponding to the cell where result was designated to be stored in M.
2. If the step is an $\stackrel{?}{=} 0$ test, then evaluate the polynomial corresponding to the cell whose value is to be tested at the given input $x_1, \ldots, x_n \in \{0,1\}$ modulo p_1. If the result is zero treat the test as affirmative, else in the negative.

We analyse the space of M' on a given input $x_1, \ldots, x_n \in \{0,1\}$. Consider a cell c of M. Let $g_c = g_c(x_1, \ldots, x_n, y_1)$ be the polynomial representing the value stored at cell c at a fixed point of time in the computation. Note that degree of y_1 in g_c at most $d-1$. Suppose $g_c = f_0 + f_1 y_1 + f_2 y_1^2 + \cdots + f_{d-1} y_1^{d-1} \mod p_1$. We have $S_{\mathsf{weak}}(f_i) \le S_{\mathsf{weak}}(g_c)$ for $0 \le i \le d-1$. The overall weak space requirement of M' is bounded by $d \cdot WSpace_M(n) = ds(n) = O(\log n)$.

For the case when $k > 1$, Let $\mathbb{G} = \mathbb{Q}(\beta_1, \ldots, \beta_k)$ be the extension field of $\mathbb{Q}$ obtained by adding $\beta_1, \ldots, \beta_k$. Clearly $\mathbb{G}$ is a finite extension of $\mathbb{Q}$. By the primitive element theorem [19], there is a $\beta \in \mathbb{F}$ such that $\mathbb{Q}(\beta) = \mathbb{G}$. Let p be the minimal polynomial for β of degree σ

with coefficients from $\mathbb{Q}$. Let $p_1(y), \ldots, p_k(y)$ be univariate polynomials of minimum degree such that $p_i(\beta) = \beta_i$, and let Δ be the maximum of degrees of p_is.

Consider an input $x_1, \ldots, x_n \in \{0, 1\}$ and a cell c of M. Suppose $g_c = g_c(x_1, \ldots, x_n, y_1, y_2, \ldots, y_k)$ is the polynomial representing the value stored at the cell c at any fixed point of time in the computation. Let d be the degree of g_c and $g_c = \sum_{\delta \in \mathbb{N}^k} f_\delta \prod_{j=1}^{k} y_j^{\delta_j}$, where f_δ is a polynomial of degree at most $d - \sum_i \delta_i$ in $x_1, \ldots, x_n$. Let

$$g'_c = g_c(x_1, \ldots, x_n, p_1(y), \ldots, p_k(y)) = \sum_{\delta \in \mathbb{N}^k} f_\delta \prod_{j=1}^{k} p_j(y)^{\delta_j} = \sum_{i=0}^{d} g'_i(x_1, \ldots, x_n) y^i.$$

$$\text{Note that, } S_{\mathsf{weak}}(g_c) = \sum_{\delta \in \mathbb{N}^k} S_{\mathsf{weak}}(f_\delta)(\sum_i \log \delta_i). \tag{1}$$

We first bound $S_{\mathsf{weak}}(g'_c \mod p)$. For $\delta \in \mathbb{N}^k$ with $\sum_i \delta_i \leq d$, let $q_\delta = \prod_{j=1}^{k} p_i(y)^{\delta_i}$. q_δ is a polynomial of degree at most $d\Delta$. Then $g'_i = \sum_{\delta : \mathsf{coeff}_{q_\delta}(y^i) \neq 0} f_\delta$, thus the number of bits required to store g'_i is bounded by $\sum_{\delta : \mathsf{coeff}_{q_\delta}(y^i) \neq 0} S_{\mathsf{weak}}(f_\delta)$. Since q_δ is of degree at most $d\Delta$ and hence $S_{\mathsf{weak}}(g'_i)$ can be dependent on d. However, $q_\delta \mod p$ is a polynomial of degree at most $\sigma - 1$ and hence any given f_δ will be a summand for at most σ many g'_is. Therefore, $S_{\mathsf{weak}}(g'_c \mod p)$ is at most $\sigma \cdot S_{\mathsf{weak}}(g_c)$.

To conclude the argument for Case 2, we describe the simulation of the machine M': M' simulates M as in the case when $k = 1$ by storing the polynomials $g'_c \mod p$ explicitly, i.e., it stores the polynomials $g'_i \mod p$. The number of bits required to store g'_c is bounded by $S_{\mathsf{weak}}(g'_c)$ which in turn is bounded by $(\sigma + 1) S_{\mathsf{weak}}(g_c)$. Now the simulation is done as in the case $k = 1$.

Case 3: M uses transcendental constants. Let γ be a transcendental number. Then for any polynomial p with integer coefficients, we have $p(\gamma) \neq 0$. Thus, for any cell c of M and for any $x_1, \ldots, x_n \in \{0, 1\}$, $g_c(x_1, \ldots, x_n, \gamma) = 0$ if and only if $g_c(x_1, \ldots, x_n, y) \equiv 0$. The simulation of M by M' can be done the same fashion as in Case 2, except that the polynomials g_c are stored as they are. Suppose $g_c(x_1, \ldots, x_n, y) = \sum_{i=0}^{d} f_i y^i$, then $S_{\mathsf{weak}}(g_c) = \sum_i S_{\mathsf{weak}}(f_i) \log i$, there fore the space required to store g_c by storing f_i's explicitly is bounded by $S_{\mathsf{weak}}(g_c)$, M' requires space at most $O(s(n)) = O(\log n)$. Now, consider the case when M uses more than one transcendental constants, and let $\gamma_1, \ldots, \gamma_k$ be the constants used by M that are transcendental. Suppose $t \leq k$ is such that γ_i is transcendental in $\mathbb{Q}(\gamma_1) \cdots (\gamma_{i-1})$ (where $\mathbb{Q}(\gamma_1)$ is the field extension of $\mathbb{Q}$ that contains γ_1) for $i \leq t$ and γ_j is algebraic over $\mathbb{G} = ((\mathbb{Q}(\gamma_1)) \cdots)(\gamma_t)$ for $j \geq t + 1$. By the primitive element theorem, let γ be such that $\mathbb{G}(\gamma) = \mathbb{G}(\gamma_{t+1}, \ldots, \gamma_k)$. Let $p_i(y)$ be a polynomial over $\mathbb{G}$ of minimal degree such that $\gamma_i = p_i(\gamma)$ for $t + 1 \leq i \leq k$. Now the simulation of M by M' can be done as in Case 2, however, the only difference is polynomials p_i can have rational functions over $\gamma_1, \ldots, \gamma_t$ as coefficients. However, any coefficient of p_i can be written as an evaluation of fraction of polynomials of constant degree over t variables, hence contributing a constant factor in the overall space requirement. Thus, for any cell c of M at any point of computation on a given input $x_1, \ldots, x_n \in \{0, 1\}$ can be represented as a polynomial $g_c(x_1, \ldots, x_n, y) \mod p$ over $\mathbb{G}$. By the observations in Case 2, and the fact that any fixed element in $\mathbb{G}$ can be represented in constant space, the overall space required by M' to simulate M is bounded by $O(\Gamma \cdot s(n)) = O(\log(n))$ where Γ is a constant that depends on k , the maximum degree of the polynomials $p_{t+1}, \ldots, p_k$ and the number bits required to represent the coefficients of these polynomials as rational functions over $\mathbb{Q}$ in $\gamma_1, \ldots, \gamma_t$. $\square$

However, we are unable to show the converse of the above theorem, i.e., the question $\mathsf{DLOG} \subseteq \mathsf{LOGSPACE_W}$? remains open. The main difficulty is, we can easily construct deterministic log-space bounded machines that evaluate non-sparse polynomials such as the elementary symmetric polynomials over a Boolean input.

4 Weak space lower bounds

In this section we exhibit languages in $\mathbb{F}^*$ that are not in $\mathsf{LOGSPACE_W}$. We begin with a simple structural observations on the languages in $\mathsf{SPACE_W}(s)$ for any non-decreasing function s.

Lemma 2. *Let $L \in \mathsf{SPACE_W}(s)$, then for every $n > 0$, there exist $t \geq 1$ and polynomials $f_{i,j}$, $1 \leq i \leq t$, $1 \leq j \leq m_i$, $g_{i,j}$ and $1 \leq i \leq t$, $1 \leq j \leq m_i$ in $\mathbb{Z}[x_1, \ldots, x_n]$ such that:*

1. *$S_{\mathsf{weak}}(f_{i,j}) \leq s(n)$, for every $1 \leq i \leq t_1$, $1 \leq j \leq m_i$; and*
2. *$S_{\mathsf{weak}}(g_{i,j}) \leq s(n)$, for every $1 \leq i \leq t_2$, $1 \leq j \leq m_i$; and*
3. *$L \cap \mathbb{F}^n = \bigcup_{i=1}^{t} \bigcap_{j=1}^{m_i} [f_{i,j} = 0] \cap \bigcap j = 1^{m_i} [g_{i,j} \neq 0]$.*

Proof. Let $L \in \mathsf{SPACE_W}(s)$, and M be an s weak space bounded BSS machine accepting L. On any input $x \in \mathbb{F}^n$, let T be the computation tree of M on an input of length n, where the branches represent the possibilities after each test. Note that every node t in T corresponds to a test $f \stackrel{?}{=} 0$, and the polynomial f depends only on the path from root of the tree to t and not on the actual input. Then any leaf ℓ in T represents a set of the form $\bigcap_{j=1}^{m} [f_j = 0] \cap \bigcap j = 1^m [g_j \neq 0]$ for some m and $f_1, \ldots, f_m$, $g_1, \ldots, g_m$ that depend only on n and ℓ. Since the $f_i's$ and $g_i's$ are polynomials computed by the machine, they satisfy the required space bound. Taking union of the sets of inputs corresponding to all accepting leaves of T proves the required result.

Definition 10. *For $n \geq 0, d \leq n$, let*

$$S_{n,d} \stackrel{\mathsf{def}}{=} \{(a_1, \ldots, a_n) \in \mathbb{F}^n \mid \mathsf{sym}_{n,d}(a_1, \ldots, a_n) = 0\}$$

i.e., the hyper surface defined by the n-variate elementary symmetric polynomial of degree d. For $d = d(n) \leq n$ define the language: $L^{(d)} \stackrel{\mathsf{def}}{=} \bigcup_{n \geq 0} S_{n,d(n)}$.

Theorem 3. *For any constant $c > 0$ $L^{(n/2)} \notin \mathsf{SPACE_W}(n^c)$.*

Proof. We argue for the case $\mathbb{F} = \mathbb{C}$. An exactly similar argument is applicable to the case when $\mathbb{F} = \mathbb{R}$. For any $c > 0$ consider an arbitrary language $L' \in \mathsf{SPACE_W}(n^c)$. Then, for every $n \geq 1$, there are n-variate polynomials $f_{i,j}$, $1 \leq i \leq t$, $1 \leq j \leq m_i$, $g_{i,j}$, $1 \leq i \leq t$, $1 \leq j \leq m_i$ in $\mathbb{Z}[x_1, \ldots, x_n]$ as promised by Lemma 2. Let

$$V_i \stackrel{\mathsf{def}}{=} \bigcap_{j=1}^{m_i} [f_{i,j} = 0]; \quad W_i \stackrel{\mathsf{def}}{=} \bigcap_{j=1}^{m_i} [g_{i,j} \neq 0]; \text{ and } T_i \stackrel{\mathsf{def}}{=} V_i \cap W_i.$$

Then we have $L' \cap \mathbb{C}^n = \bigcup_{i=1}^{t} T_i$. We argue that for large enough n, $\bigcup_{i=1}^{t} T_i \neq S_{n,n/2}$ and hence conclude $L' \neq L^{(n/2)}$. Let $\widehat{T_i}$ denote the Zariski closure of the set T_i in $\mathbb{C}^n$, i.e, the smallest algebraic variety containing T_i. Proof is by contradiction. Suppose that $\bigcup_{i=1}^{t} T_i = S_{n,n/2}$. As $S_{n,n/2}$ is a closed set in the Zariski topology over $\mathbb{C}^n$, we have $T_i \subseteq \widehat{T_i} \subseteq S_{n,n/2}$ and hence $\bigcup_{i=1}^{t} \widehat{T_i} = S_{n,n/2}$. Then, there should be an i such that $\widehat{T_i} = S_{n,n/2}$, for, $S_{n,n/2}$ is an irreducible algebraic variety. Now there are two cases:

Case 1: $V_i = \mathbb{C}^n$. In this case, $T_i = W_i$ i.e., an open set in the Zariski topology. Since $\mathbb{C}^n$ is dense in the Zariski topology, closure of any open set is in fact $\mathbb{C}^n$ itself. Therefore, $\widehat{T_i} = \mathbb{C}^n \neq S_{n,n/2}$, hence a contradiction.

Case 2: $V_i \neq \mathbb{C}^n$. Then we have $T_i = V_i \cap W_i \subseteq V_i$, therefore $S_{n,n/2} = \widehat{T_i} \subseteq V_i = \bigcap_{j=1}^{m_i} [f_{ij} = 0]$. It is enough to argue that $S_{n,n/2}$ is not contained in any of the varieties $[f_{i,j} = 0]$. Suppose $S_{n,n/2} \subseteq [f_{i,j} = 0]$ for some $1 \leq j \leq m_i$. Since $\mathsf{sym}_{n,n/2}$ is an irreducible polynomial, we have $\mathsf{sym}_{n,n/2} | f_{i,j}$. By Corollary 2, the number of monomials in $f_{i,j}$ is $n^{\omega(1)}$. However, by Lemma 2, the number of monomials in f_{ij} is at most $O(n^c)$, obtaining a contradiction for large enough n. Thus $S_{n,n/2} \not\subseteq [f_{i,j} = 0]$ for any $1 \leq j \leq m_i$ which in turn implies $S_{n,n/2} \not\subseteq V_i$ and hence $S_{n,n/2} \not\subseteq \widehat{T_i}$. Thus in both of the cases above we obtain a contradiction, as a result we have $S_{n,n/2} \neq \bigcup_{i=1}^{t} T_i$. Thus $L' \neq L^{(n/2)}$ as required.

As an immediate corollary we have:

Corollary 1. $\mathsf{NC}^1_{\mathbb{F}} \not\subseteq \mathsf{PSPACE}_{\mathsf{W}}$

Proof. It is known that $\mathsf{sym}_{n,d}$ is computable by polynomial size arithmetic circuits of logarithmic depth [24] and hence $L^{(d)} \in \mathsf{NC}^1_{\mathbb{F}}$. The result follows.

Now, to complete the proof of Theorem 3, we need to prove Corollary 2. This is done in the next section using the properties of Newton's polytope of elementary symmetric polynomials.

5 Polynomials divisible by elementary symmetric polynomials

Let g be a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. In this section we prove that, for any polynomial f which is a polynomial multiple of g, the number of monomials of f are lower bounded by number of vertices of Newton polytope of g. As an implication we get an exponential lower bound on number of monomials of any polynomial multiple of $\mathsf{sym}_{n,d}$. The key step in the proof is a simple Lemma which lower bounds number of vertices of convex polytope R in terms of number of vertices of convex polytopes P and Q when R is Minkowski sum of P and Q. We begin with definition of Minkowski sum.

Definition 11 (Minkowski sum). *For $A, B \subseteq \mathbb{R}^d$, Minkowski sum of A and B (denoted by $A \oplus B$) is defined as $A \oplus B = \{a + b | a \in A, b \in B\}$.*

Minkowski sums of convex sets have been extensively studied in mathematics literature, and has interesting applications in complexity theory, see for example [21, 8, 15]. The next proposition shows that the Minkowski sum of two convex polytopes is a convex polytope and every vertex of resulting polytope can be uniquely expressed as sum of vertices of the two polytopes. In fact, a more general statement about unique decomposition of a face (of any dimension) of Minkowski sum of convex polytopes into faces of individual polytopes holds true, see for example [9], [21].

Proposition 4 *If $P, Q \subseteq \mathbb{R}^d$ are convex polytopes then the Minkowski sum of P and Q is a convex polytope $P \oplus Q = conv(\{p+q | p \in ver(P), q \in ver(Q)\})$ and for every vertex $r \in ver(P \oplus Q)$ there exist* unique *$p \in P, q \in Q$ such that $r = p+q$, moreover $p \in ver(P), q \in ver(Q)$.*

Proof. Let $ver(P) = \{p_1, \ldots, p_m\}$ and $ver(Q) = \{q_1, \ldots, q_n\}$. First we prove $R = conv(\{p + q | p \in ver(P), q \in ver(Q)\}) \subseteq P \oplus Q$. Let $v \in R$. So v can be written as convex combination of points $p_i + q_j$ for $i \in [m], j \in [n]$.

$$v = \sum_{\ell} \lambda_\ell (p_{i_\ell} + q_{j_\ell}) \text{ for } 0 \leq \lambda_\ell \leq 1, \sum_{\ell} \lambda_\ell = 1$$

where $p_{i_\ell} \in ver(P)$ and $q_{j_\ell} \in ver(Q)$. So for $v_p = \sum_\ell \lambda_\ell p_{i_\ell}$ and $v_q = \sum_\ell \lambda_\ell q_{j_\ell}$, $v = v_p + v_q$, where $v_p \in P$ and $v_q \in Q$. Which imply $v \in P \oplus Q$.

To see the other inclusion, consider point $v_p + v_q \in P \oplus Q$ for $v_p = \sum_\ell \lambda_\ell p_\ell, 0 \leq \lambda_\ell \leq 1, \sum_\ell \lambda_\ell = 1$ and $v_q = \sum_\ell \lambda_\ell q_\ell, 0 \leq \lambda'_\ell \leq 1, \sum_\ell \lambda'_\ell = 1$. To prove that $v \in R$ we need to express v as a convex combination of points $(p_i + q_j)$'s. Consider the following sum

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i \lambda'_j (p_i + q_j) &= \sum_{i=1}^{m} (\lambda_i p_i \sum_{j=1}^{n} \lambda'_j + \lambda_i \sum_{j=1}^{q} \lambda'_j q_j) \\
&= \sum_{i=1}^{m} (\lambda_i p_i + \lambda_i v_q) \\
&= v_p + v_q
\end{aligned}$$

So clearly $v = v_p + v_q \in R = conv(\{p + q | p \in ver(P), q \in ver(Q)\})$.

Now we will argue that if vertex $v \in ver(P \oplus Q)$ is expressed as $v = v_p + v_q$ for $v_p \in P$ and $v_q \in Q$ then v_p and v_q must be vertices of P and Q respectively. Let $v = v_p + v_q$ for $v_p \in P$ and $v_q \in Q$ and with out loss of generality assume that v_p is not a vertex of P. So from Proposition 2 v_p can be expressed as non-trivial convex combination of $ver(P)$

$$v = \sum_{\ell=1}^{m} \lambda_\ell p_\ell, \quad 0 \leq \lambda_\ell < 1, \quad \sum_{\ell} \lambda_l = 1$$

Let $v_q = \sum_{\ell=1}^{n} \lambda'_\ell q_\ell, \quad 0 \leq \lambda'_\ell \leq 1, \quad \sum_\ell \lambda'_l = 1$. As $\lambda_\ell < 1$ for $\ell \in [m]$ and $\lambda'_\ell \leq 1$ for $\ell \in [n]$, we get $0 \leq \lambda_i \lambda'_j < 1$ for $i \in [m], j \in [n]$. we have,

$$v = v_p + v_q = \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_i \lambda'_j (p_i + q_j)$$

where $0 \le \lambda_i \lambda'_j < 1$ and $\sum_{i,j} \lambda_i \lambda'_j = 1$. So we can express $v \in ver(P \oplus Q)$ as non-trivial convex combination of $ver(P \oplus Q)$, a contradiction to Proposition 2. This shows that if vertex $v \in ver(P \oplus Q)$ is expressed as $v = v_p + v_q$ for $v_p \in P$ and $v_q \in Q$ then v_p and v_q must be vertices of P and Q respectively.

Now we will prove the uniqueness. Suppose vertex $v \in ver(P \oplus Q)$ can be expressed as sum of vertices of P and Q in two different ways, $v = v_p + v_q = v_{p'} + v_{q'}$ for $v \in ver(P \oplus Q), v_p, v_{p'} \in ver(P)$, $v_q, v_{q'} \in ver(Q)$ and without loss of generality assume that $v_p \neq v_{p'}$. We have

$$v = \frac{1}{2}(v_p + v_{p'}) + \frac{1}{2}(v_q + v_{q'}).$$

By Proposition 2 $\frac{1}{2}(v_p + v_{p'}) \notin ver(P)$. This is a contradiction as we have already proved above that for a vertex $v \in ver(P \oplus Q)$ if $v = u + w$ for $u \in P, w \in Q$ then $u \in ver(P)$ and $w \in ver(Q)$.

□

Lemma 3. *For convex polytopes $P, Q \subseteq \mathbb{R}^d$,*

$$|ver(P \oplus Q)| \ge max(|ver(P)|, |ver(Q)|).$$

Proof. Let $ver(P) = \{p_1, p_2, \ldots, p_m\}$, $ver(Q) = \{q_1, q_2, \ldots, q_n\}$ and $m \ge n$. To the contrary assume that $|ver(P \oplus Q)| < m$ and let $R = P \oplus Q$ and $ver(R) = \{r_1, r_2, \ldots, r_t\}$, where $t < m$. From Proposition 4, for $\ell \in [t]$ every vertex $r_\ell \in ver(R)$ can be *uniquely* expressed as $r_\ell = p_{i_\ell} + q_{j_\ell}$ where $p_{i_\ell} \in ver(P)$ and $q_{j_\ell} \in ver(Q)$. But as $t = |ver(R)| < m = |ver(P)|$, there must be a vertex $p' \in ver(P)$ which plays no role in determining any vertex of $P \oplus Q$, that is, every $r_\ell \in ver(P \oplus Q)$ can be expressed as $r_\ell = p_{i_\ell} + q_{j_\ell}$ where $p_{i_\ell} \in ver(P) \setminus \{p'\}$ and $q_{j_\ell} \in ver(Q)$. Without loss of generality assume that $p' = p_1$. Since p_1 is a vertex of P, there exist a valid linear inequality $\langle v, p_1 \rangle \le k$,$k \in \mathbb{R}, v \in \mathbb{R}^d$ such that $\langle v, p_1 \rangle = k$ and for any $x \in P \setminus \{p_1\}$, $\langle v, x \rangle < k$. Let $q \in Q$ such that $\langle v, y \rangle \le \langle v, q \rangle = k'$, $k' \in \mathbb{R}$ for any $y \in Q$. Let $z = p_1 + q \in P \oplus Q$.

From Proposition 3 we know that $R = P \oplus Q = conv(ver(P \oplus Q))$. So the point $z \in P \oplus Q$ can be expressed as $z = \sum_{\ell=1}^{t} \lambda_\ell (p_{i_\ell} + q_{j_\ell})$ where $\lambda_\ell \ge 0, \sum_\ell \lambda_\ell = 1$ where $p_{i_\ell} \in ver(P) \setminus \{p_1\}$ and $q_{j_\ell} \in ver(Q)$. Let $z_P = \sum_{\ell=1}^{t} \lambda_\ell p_{i_\ell} \in P$ and $z_Q = \sum_{\ell=1}^{t} \lambda_\ell q_{j_\ell} \in Q$. So we get $z = z_P + z_Q = p_1 + q$. First we argue that $p_1 \neq z_P$. Assume $p_1 = z_P = \sum_{\ell=1}^{t} \lambda_\ell p_{i_\ell}$, where $p_{i_\ell} \in ver(P) \setminus \{p_1\}$. Clearly if $\lambda_\ell = 1$ for some $\ell \in [t]$ then $\lambda_i = 0$ for $i \in [t] \setminus \{\ell\}$ and we get $p_1 = p_{i_\ell}$ but that is not possible as $p_{i_\ell} \in ver(P) \setminus \{p_1\}$. So we can express a vertex p_1 of P as a nontrivial convex combination of $p_{i_1}, p_{i_2}, \ldots p_{i_t} \in ver(P) \setminus \{p_1\}$. A contradiction to Proposition 2. So $p_1 \neq z_P$.

We know that $\langle v, p_1 \rangle = k$ and for any $x \in P \setminus \{p_1\}$,$\langle v, p_1 \rangle < k$. In particular, $\langle v, z_P \rangle < k$. Also, by choice of q we have $\langle v, y \rangle \le \langle v, q \rangle$ for $y \in Q$. As a result we get $\langle v, z_P \rangle + \langle v, z_Q \rangle < \langle v, p_1 \rangle + \langle v, q \rangle$. A contradiction, since $z = z_P + z_Q = p_1 + q$.

Now we recall the notion of Newton's polytope of polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. Let f be a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. Let $f_{(\alpha_1, \alpha_2, \ldots, \alpha_n)}$ denote the coefficient of the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ in f,

$$f = \sum f_{(\alpha_1, \alpha_2, \ldots, \alpha_n)} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}.$$

A vector $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{R}^n$ is called as an exponent vector of the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ of f. The Newtone polytope of f is defined as the convex-hull of set of exponent vectors $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ in $\mathbb{R}^n$ for which $f_{(\alpha_1,\alpha_2,\ldots,\alpha_n)} \neq 0$. The Newton polytope of f is denoted by P_f.

For a polynomial f, let $mon(f)$ denote the set of monomials with non-zero coefficient in f. Following Lemma is from [8]. As per [8] a more general version of Lemma 4 appears in [20]. We include the proof of Lemma 4 in the Appendix.

Lemma 4. *([20]) Let $f, g, h \in \mathbb{F}[x_1, \ldots, x_n]$ with $f = gh$ then $P_f = P_g \oplus P_h$.*

Proof. First we will prove the inclusion $P_f \subseteq P_g \oplus P_h$. Let $\gamma = (\gamma_1, \ldots, \gamma_n) \in ver(P_f)$. So the monomial $m = x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n} \in mon(f)$. Since, $f = gh$, there exists monomials $m1 = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \in mon(g)$ and $m_2 = x_1^{\beta_1} x_2^{\beta_2} \ldots x_n^{\beta_n} \in mon(h)$ such that $m = m_1 m_2$. So clearly, for $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ we have $\alpha \in P_g, \beta \in P_h$ and $\gamma = \alpha + \beta$. That implies $\gamma \in P_g \oplus P_h$. So every vertex of P_f is in $P_g \oplus P_h$. By Proposition 3, $P_f = conv(ver(P_f))$. By definition of convex-hull and Lemma 1 it clearly implies $P_f \subseteq P_g \oplus P_h$.

Now we prove that $P_g \oplus P_h \subseteq P_f$. It is enough to prove that $ver(P_g \oplus P_h) \subseteq P_f$, as the desired inclusion will then follow from Proposition 3 and Lemma 1. Let $v_f = (e_1, \ldots, e_n) \in ver(P_f)$. By Proposition 4, there exist *unique* $v_g \in ver(P_g)$ and $v_h \in ver(P_h)$ such that $v_f = v_g + v_h$. So there exist unique monomials $m_1 \in mon(g)$ and $m_2 \in mon(h)$ such that $x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n} = m_1 m_2$. Since the monomial $x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$ can be uniquely generated as a product of monomial from g and h, it can not be cancelled off and will be present in $mon(f)$. As a result $v_f \in P_f$. This completes the proof. $\square$

The Proof of Theorem 4 below follows immediately from Lemma 3, 4.

Theorem 4. *Let f, g, h be nonzero polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ with $f = gh$ then $|mon(f)| \geq max(|ver(P_g)|, |ver(P_h)|)$.*

Corollary 2. *For any nonzero polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ let $f = g \cdot sym_{n,\frac{n}{2}}$ then $|mon(f)| \in 2^{\Omega(n)}$.*

Proof. For a subset $S \subseteq [n]$ of size k, let v_S be the exponent vector corresponding to the monomial $\Pi_{j \in S} x_j \in mon(sym_{n,k})$, i.e. $v_S(j) = 1$ for $j \in S$ and $v_S(j) = 0$ otherwise. We will argue that every vector v_S for $S \subseteq [n]$, $|S| = k$ is a vertex of the Newton polytope of $sym_{n,k}$. Suppose v_S is not a vertex for some $S \subseteq [n]$. So by Proposition 2 v_S can be expressed as non-trivial convex combination of exponent vectors, $v_S = \sum_{T \subset [n], |T|=k} \lambda_T v_T$ where $0 \leq \lambda_T < 1$, $\sum_T \lambda_T = 1$. Consider any U such that $\lambda_U > 0$ and let $i \in U \setminus S$. Since $\lambda_T \geq 0$, clearly $v_S(i) = \sum_{T \subset [n], |T|=k} \lambda_T v_T(i) \geq \lambda_U v_U(i) > 0$. A contradiction, as by choice of i, $v_S(i) = 0$. So every vector v_S for $S \subseteq [n]$, $|S| = k$ is a vertex of Newton polytope of $sym_{n,k}$. So $|ver(P_h)| = \binom{n}{\frac{n}{2}} \in 2^{\Omega(n)}$ for $h = sym_{n,\frac{n}{2}}$. The desired result follows from Theorem 4.

6 Conclusions and Future directions

Our study reveals that obtaining a good notion of space for the BSS model of algebraic computation still remains a challenging task. We showed that the Boolean part of $\mathsf{LOGSPACE}_w$ is contained in DLOG, however the converse containment is unlikely and it remains open to show that $\mathsf{DLOG} \not\subset \mathsf{LOGSPACE}_w$.

Acknowledgements We thank the anonymous reviewers for an earlier version of the paper for suggestions that helped to improve the presentation of proofs.

References

1. Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, January 2009.
2. Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation*. Springer, 1997.
3. Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, 1989.
4. Felipe Cucker. P_R != NC_R. *J. Complexity*, 8(3):230–238, 1992.
5. Felipe Cucker and Dima Grigoriev. On the power of real turing machines over binary inputs. *SIAM Journal on Computing*, 26(1):243–254, 1997.
6. Paulin Jacobé de Naurois. A Measure of Space for Computing over the Reals. In *CiE*, pages 231–240, 2006.
7. Hervé Fournier and Pascal Koiran. Are lower bounds easier over the reals? In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 507–513, New York, NY, USA, 1998. ACM.
8. Shuhong Gao. Absolute irreducibility of polynomials via newton polytopes. *Journal of Algebra*, 237(1):501–520, 1997.
9. B. Gruenbaum. *Convex Polytopes*. Interscience Publisher, 1967.
10. Pascal Koiran. Computing over the reals with addition and order. *Theoretical Computer Science*, 133(1):35–47, 1994.
11. Pascal Koiran. Elimination of constants from machines over algebraically closed fields. *J. Complexity*, 13(1):65–82, 1997.
12. Pascal Koiran. A weak version of the blum, shub, and smale model. *J. Comput. Syst. Sci.*, 54(1):177–189, 1997.
13. Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the complex field. *Theor. Comput. Sci.*, 410(50):5244–5251, 2009.
14. Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the reals. *Computational Complexity*, 18(4):551–575, 2009.
15. Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A $$\tau $$ τ -conjecture for newton polygons. *Foundations of Computational Mathematics*, 15(1):185–197, 2015.
16. Meena Mahajan and B. V. Raghavendra Rao. Small space analogues of valiant's classes and the limitations of skew formulas. *Computational Complexity*, 22(1):1–38, 2013.
17. Klaus Meer and Christian Michaux. A survey on real structural complexity theory. *Bull. Belg. Math. Soc. Simon Stevin*, 4(1):113–148, 1997.
18. 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.
19. P. Morandi. *Field and Galois Theory*. Graduate Texts in Mathematics. Springer, 1996.
20. Alexander Markowich Ostrowski. On multiplication and factorization of polynomials, i. lexicographic ordering and extreme aggregates of terms. *Aequations Math*, 13:201–228, 1975.
21. R. Schneider. *Convex bodies: the Brunn-Minkowski theory*. Cambridge University Press, 1993.
22. Igor R Shafarevich. *Basic algebraic geometry; 3rd ed.* Springer, Berlin, 2013.
23. Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.
24. Iddo Tzamaret. *Studies in Algebraic and Propositional Proof Complexity*. PhD thesis, Tel Aviv University, 2008.
25. Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
26. G. M Ziegler. *Lectures on Polytopes*. Sprnger verlag, 1995.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il