VPSPACE and a transfer theorem over the complex field

Pascal Koiran and Sylvain Perifel

LIP[⋆], École Normale Supérieure de Lyon.
[Pascal.Koiran,Sylvain.Perifel]@ens-lyon.fr

November 16, 2006

Abstract. We extend the transfer theorem of [13] to the complex field. That is, we investigate the links between the class VPSPACE of families of polynomials and the Blum-Shub-Smale model of computation over $\mathbb{C}$. Roughly speaking, a family of polynomials is in VPSPACE if its coefficients can be computed in polynomial space. Our main result is that if (uniform, constant-free) VPSPACE families can be evaluated efficiently then the class $\mathrm{PAR}_{\mathbb{C}}$ of decision problems that can be solved in parallel polynomial time over the complex field collapses to $\mathrm{P}_{\mathbb{C}}$. As a result, one must first be able to show that there are VPSPACE families which are hard to evaluate in order to separate $\mathrm{P}_{\mathbb{C}}$ from $\mathrm{NP}_{\mathbb{C}}$, or even from $\mathrm{PAR}_{\mathbb{C}}$.

Keywords: computational complexity, algebraic complexity, Blum-Shub-Smale model, Valiant's model.

[⋆] UMR 5668 ENS Lyon, CNRS, UCBL, INRIA.

1 Introduction

Two main categories of problems are studied in algebraic complexity theory: evaluation problems and decision problems. A typical example of an evaluation problem is the evaluation of the permanent of a matrix, and it is well known that the permanent family is complete for the class VNP of “easily definable” polynomial families [17]. Deciding whether a system of polynomial equations has a solution over $\mathbb{C}$ is a typical example of a decision problem. This problem is NP-complete in the Blum-Shub-Smale model of computation over the complex field [1,2].

The main purpose of this paper is to provide a transfer theorem connecting the complexity of evaluation and decision problems. This paper is therefore in the same spirit as [12] and [13]. In the present paper we work with the class of polynomial families VPSPACE introduced in [13]. Roughly speaking, a family of polynomials (of possibly exponential degree) is in VPSPACE if its coefficients can be evaluated in polynomial space. For instance, it can be shown that resultants of systems of multivariate polynomial equations form a VPSPACE family, see [13]. The main result in [13] was that if (uniform, constant-free) VPSPACE families can be evaluated efficiently then the class $\mathrm{PAR}_{\mathbb{R}}$ of decision problems that can be solved in parallel polynomial time over the real numbers collapses to $\mathrm{P}_{\mathbb{R}}$.

Here we extend this result to the complex field $\mathbb{C}$. At first glance the result seems easier because the order $\leq$ over the reals does not have to be taken into account. The result of [13] indeed makes use of a clever combinatorial lemma of [9] on the existence of a vector orthogonal to roughly half a collection of vectors. More precisely, it relies on the constructive version of this lemma [5]. On the complex field, we do not need this construction.

But the lack of an order over $\mathbb{C}$ makes another part of the proof more difficult. Indeed, over $\mathbb{R}$ testing whether a point belongs to a real variety is done by testing whether the sum of the squares of the polynomials is zero, a trick that cannot be used over the complex field. Hence one of the main technical developments of this paper is to explain how to decide with a small number of tests whether a point is in the complex variety defined by an exponential number of polynomials. This enables us to follow the nonconstructive proof of [11] for our transfer theorem.

Therefore, the main result of the present paper is that if (uniform, constant-free) VPSPACE families can be evaluated efficiently then the class $\mathrm{PAR}_{\mathbb{C}}$ of decision problems that can be solved in parallel polynomial time over the complex field collapses to $\mathrm{P}_{\mathbb{C}}$ (this is precisely stated in Theorem 2). The class $\mathrm{PAR}_{\mathbb{C}}$ plays roughly the same role in the theory of computation over the complex field as PSPACE in discrete complexity theory. In particular, it contains $\mathrm{NP}_{\mathbb{C}}$ [1] (but the proof of this inclusion is much more involved than in the discrete case). It follows from our main result that in order to separate $\mathrm{P}_{\mathbb{C}}$ from $\mathrm{NP}_{\mathbb{C}}$, or even from $\mathrm{PAR}_{\mathbb{C}}$, one must first be able to show that there are VPSPACE families which are hard to evaluate. This seems to be a very challenging lower bound problem, but it is still presumably easier than showing that the permanent is hard to evaluate.

Organization of the paper. We first recall in Section 2 some usual notions and notations concerning algebraic complexity (Valiant’s model, the Blum-Shub-Smale model) and quantifier elimination. The class VPSPACE is defined in Section 3 and some properties proved in [13] are given. Section 4 explains how to decide with a polynomial number of VPSPACE tests whether a point belongs to a variety. The main difficulty here is that the variety is given as a union of an exponential number of varieties, each defined by an exponential number of polynomials. Finally, Section 5 is devoted to the proof of the transfer theorem. Sign conditions are the main tool in this section. We show that $\mathrm{PAR}_{\mathbb{C}}$ problems are decided in polynomial time if we allow uniform $\mathrm{VPSPACE}^0$ tests. The transfer theorem follows as a corollary.

2 Notations and Preliminaries

2.1 The Blum-Shub-Smale Model

In contrast with boolean complexity, algebraic complexity deals with other structures than $\{0, 1\}$. In this paper we will focus on the complex field $(\mathbb{C}, +, -, \times, =)$. Although the original definitions of Blum, Shub and Smale [2,1] are in terms of uniform machines, we will follow [16] by using families of algebraic circuits to recognize languages over $\mathbb{C}$, that is, subsets of $\mathbb{C}^\infty = \bigcup_{n\geq 0} \mathbb{C}^n$.

An algebraic circuit is a directed acyclic graph whose vertices, called gates, have indegree 0, 1 or 2. An input gate is a vertex of indegree 0. An output gate is a gate of outdegree 0. We assume that there is only one such gate in the circuit. Gates of indegree 2 are labelled by a symbol from the set $\{+, -, \times\}$. Gates of indegree 1, called test gates, are labelled "$= 0$?". The size of a circuit C, in symbols $|C|$, is the number of vertices of the graph.

A circuit with n input gates computes a function from $\mathbb{C}^n$ to $\mathbb{C}$. On input $\bar{u} \in \mathbb{C}^n$ the value returned by the circuit is by definition equal to the value of its output gate. The value of a gate is defined in the usual way. Namely, the value of input gate number i is equal to the i-th input u_i. The value of other gates is then defined recursively: it is the sum of the values of its entries for a $+$-gate, their difference for a $-$-gate, their product for a $\times$-gate. The value taken by a test gate is 0 if the value of its entry is $\neq 0$ and 1 otherwise. Since we are interested in decision problems, we assume that the output is a test gate: the value returned by the circuit is therefore 0 or 1.

The class $\mathrm{P}_\mathbb{C}$ is the set of languages $L \subseteq \mathbb{C}^\infty$ such that there exists a tuple $\bar{a} \in \mathbb{C}^p$ and a P-uniform family of polynomial-size circuits (C_n) satisfying the following condition: C_n has exactly $n + p$ inputs, and for any $\bar{x} \in \mathbb{C}^n$, $\bar{x} \in L \Leftrightarrow C_n(\bar{x}, \bar{a}) = 1$. The P-uniformity condition means that C_n can be built in time polynomial in n by an ordinary (discrete) Turing machine. Note that $\bar{a}$ plays the role of the machine constants of [1,2].

As in [4], we define the class $\mathrm{PAR}_\mathbb{C}$ as the set of languages over $\mathbb{C}$ recognized by a PSPACE-uniform family of algebraic circuits of polynomial depth (and possibly exponential size), with constants $\bar{a}$ as for $\mathrm{P}_\mathbb{C}$. Note at last that we could also define similar classes without constants $\bar{a}$. We will use the superscript 0 to denote these constant-free classes, for instance $\mathrm{P}^0_\mathbb{C}$ and $\mathrm{PAR}^0_\mathbb{C}$.

We end this section with a theorem on the first-order theory of the complex numbers: quantifiers can be eliminated without much increase of the coefficients and degree of the polynomials. We give a weak version of the result of [8]: in particular, we do not need efficient elimination algorithms. Note that the only allowed constants in our formulae are 0 and 1 (in particular, only integer coefficients can appear). For notational consistency with the remainding of the paper, we denote by 2^s, 2^d and 2^{2^M} the number of polynomials, their degree and the size of their coefficients respectively. This will simplify the calculations and emphasize that s, d and M will be polynomial. Note furthermore that the polynomial $p(n, s, d)$ in the theorem is independent of the formula ϕ.

Theorem 1. *Let ϕ be a first-order formula over the structure $(\mathbb{C}, 0, 1, +, -, \times, =)$ of the form $\forall\bar{x}\psi(\bar{x})$, where $\bar{x}$ is a tuple of n variables and ψ a quantifier-free formula where 2^s polynomials occur. Suppose that their degrees are bounded by 2^d and their coefficients by 2^{2^M} in absolute value.*

There exists a polynomial $p(n, s, d)$, independent of ϕ, such that the formula ϕ is equivalent to a quantifier-free formula ψ in which all polynomials have degree less than $D(n, s, d) = 2^{p(n,s,d)}$, and their coefficients are integers strictly bounded in absolute value by $2^{2^M D(n,s,d)}$.

2.2 Valiant's Model

In Valiant's model, one computes polynomials instead of recognizing languages. We thus use arithmetic circuits instead of algebraic circuits. A book-length treatment of this topic can be found in [3].

An arithmetic circuit is the same as an algebraic circuit but test gates are not allowed. That is to say we have indeterminates $x_1, \ldots, x_{u(n)}$ as input together with arbitrary constants of $\mathbb{C}$; there are $+$, $-$ and $\times$-gates, and we therefore compute multivariate polynomials.

The polynomial computed by an arithmetic circuit is defined in the usual way by the polynomial computed by its output gate. Thus a family (C_n) of arithmetic circuits computes a family (f_n) of polynomials, $f_n \in \mathbb{C}[x_1, \ldots, x_{u(n)}]$. The class $\mathrm{VP_{nb}}$ defined in [14] is the set of families (f_n) of polynomials computed by a family (C_n) of polynomial-size arithmetic circuits, i.e., C_n computes f_n and there exists a polynomial $p(n)$ such that $|C_n| \leq p(n)$ for all n. We will assume without loss of generality that the number $u(n)$ of variables is bounded by a polynomial function of n. The subscript "nb" indicates that there is no bound on the degree of the polynomial, in contrast with the original class VP of Valiant where a polynomial bound on the degree of the polynomial computed by the circuit is required.

The class VNP is the set of families of polynomials defined by an exponential sum of VP families. More precisely, $(f_n(\bar{x})) \in \mathrm{VNP}$ if there exists $(g_n(\bar{x}, \bar{y})) \in \mathrm{VP}$ and a polynomial p such that $|\bar{y}| = p(n)$ and $f_n(\bar{x}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(n)}} g_n(\bar{x}, \bar{\epsilon})$. Note that these definitions are nonuniform. The class uniform $\mathrm{VP_{nb}}$ is obtained by adding a condition of polynomial-time uniformity on the circuit family, as in Section 2.1.

We can also forbid constants from our arithmetic circuits in unbounded-degree classes, and define constant-free classes. The only constant allowed is 1 (in order to allow the computation of constant polynomials). As for classes of decision problems, we will use the superscript 0 to indicate the absence of constant: for instance, we will write $\mathrm{VP^0_{nb}}$ (for bounded-degree classes, we are to be more careful: the "formal degree" of the circuits comes into play, see [14,15]).

3 The Class VPSPACE

The class VPSPACE was introduced in [13]. Some of its properties are given there and a natural example of a VPSPACE family coming from algebraic geometry, namely the resultant of a system of polynomial equations, is provided. In this section, after the definition we give some properties without proof and refer to [13] for further details.

3.1 Definition

We fix an arbitrary field K. The definition of VPSPACE will be stated in terms of *coefficient function.* A monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is encoded in binary by $\alpha = (\alpha_1, \ldots, \alpha_n)$ and will be written $\bar{x}^{\alpha}$.

Definition 1. *Let (f_n) be a family of multivariate polynomials with integer coefficients. The coefficient function of (f_n) is the function a whose value on input (n, α, i) is the i-th bit $a(n, \alpha, i)$ of the coefficient of the monomial $\bar{x}^{\alpha}$ in f_n. Furthermore, $a(n, \alpha, 0)$ is the sign of the coefficient of the monomial $\bar{x}^{\alpha}$. Thus f_n can be written as*

$$f_n(\bar{x}) = \sum_{\alpha} \Big((-1)^{a(n,\alpha,0)} \sum_{i \geq 1} a(n, \alpha, i) 2^{i-1} \bar{x}^{\alpha} \Big).$$

The coefficient function is a function $a : \{0,1\}^* \to \{0,1\}$ and can therefore be viewed as a language. This allows us to speak of the complexity of the coefficient function.

Definition 2. *The class* uniform $\mathrm{VPSPACE}^0$ *is the set of all families (f_n) of multivariate polynomials $f_n \in K[x_1, \ldots, x_{u(n)}]$ satisfying the following requirements:*

1. *the number $u(n)$ of variables is polynomially bounded;*

2. *the polynomials f_n have integer coefficients;*
3. *the size of the coefficients of f_n is bounded by $2^{p(n)}$ for some polynomial p;*
4. *the degree of f_n is bounded by $2^{p(n)}$ for some polynomial p;*
5. *the coefficient function of (f_n) is in* PSPACE.

We have chosen to present only uniform $\mathrm{VPSPACE}^0$, a uniform class without constants, because this is the main object of study in this paper. In keeping with the tradition set by Valiant, however, the class VPSPACE is nonuniform and allows for arbitrary constants. See [13] for a precise definition.

3.2 An Alternative Characterization and Some Properties

Let uniform VPAR^0 be the class of families of polynomials computed by a PSPACE-uniform family of constant-free arithmetic circuits of polynomial depth (and possibly exponential size). This in fact characterizes uniform $\mathrm{VPSPACE}^0$. The proof is given in [13].

Proposition 1. *The two classes* uniform $\mathrm{VPSPACE}^0$ *and* uniform VPAR^0 *are equal.*

We see here the similarity with $\mathrm{PAR}_{\mathbb{C}}$, which by definition are those languages recognized by uniform algebraic circuits of polynomial depth. But of course there is no test gate in the arithmetic circuits of uniform VPAR^0.

We now turn to some properties of VPSPACE. The following two propositions come from [13]. They stress the unlikeliness of the hypothesis that VPSPACE has polynomial-size circuits.

Proposition 2. *Assuming the generalized Riemann hypothesis (GRH),* $\mathrm{VP_{nb}} = \mathrm{VPSPACE}$ *if and only if* [P/poly = PSPACE/poly *and* VP = VNP]*. Moreover, the "if" direction holds even without GRH.*

Proposition 3. Uniform $\mathrm{VPSPACE}^0 =$ uniform $\mathrm{VP}^0_{\mathrm{nb}} \implies \mathrm{PSPACE} =$ P-uniform NC.

Remark 1. To the authors' knowledge, the separation "PSPACE $\neq$ P-uniform NC" is not known to hold (by contrast, PSPACE can be separated from logspace-uniform NC thanks to the space hierarchy theorem).

Let us now state the main result of this paper.

Theorem 2 (main theorem). *If* uniform $\mathrm{VPSPACE}^0 =$ uniform $\mathrm{VP}^0_{\mathrm{nb}}$ *then* $\mathrm{PAR}^0_{\mathbb{C}} = \mathrm{P}^0_{\mathbb{C}}$.

Note that the collapse of the constant-free class $\mathrm{PAR}^0_{\mathbb{C}}$ to $\mathrm{P}^0_{\mathbb{C}}$ implies $\mathrm{PAR}_{\mathbb{C}} = \mathrm{P}_{\mathbb{C}}$: just replace constants by new variables so as to transform a $\mathrm{PAR}_{\mathbb{C}}$ problem into a $\mathrm{PAR}^0_{\mathbb{C}}$ problem, and then replace these variables by their original values so as to transform a $\mathrm{P}^0_{\mathbb{C}}$ problem into a $\mathrm{P}_{\mathbb{C}}$ problem.

The next section is devoted to the problem of testing whether a point belongs to a variety. This problem is useful for the proof of the theorem: indeed, following [11], several tests of membership to a variety will be made; the point here is to make them constructive and efficient. The main difficulty is that the variety can be defined by an exponential number of polynomials.

4 Testing Membership to a Union of Varieties

In this section we explain how to perform in uniform $\mathrm{VPSPACE}^0$ membership tests of the form "$\bar{x} \in V$", where $V \subseteq \mathbb{C}^n$ is a variety. We begin in Section 4.1 by the case where V is given by s polynomials. In that case, we determine after some precomputation whether $\bar{x} \in V$ in $n+1$ tests.

We first need two lemmas given below in order to reduce the number of polynomials and to replace transcendental elements by integers.

Then, in Section 4.2, we deal with the case where V is given as a union of an exponential number of such varieties, as in the actual tests of the algorithm of Section 5. Determining whether $\bar{x} \in V$ still requires $n+1$ tests, but the precomputation is slightly heavier.

Let us first state two useful lemmas. Suppose a variety V is defined by $f_1, \dots, f_s$, where $f_i \in \mathbb{Z}[x_1, \dots, x_n]$. We are to determine whether $\bar{x} \in V$ with only $n+1$ tests, however big s might be. In a nonconstructive manner, this is possible and relies on the following classical lemma already used (and proved) in [11]: any $n+1$ "generic" linear combinations of the f_i also define V (the result holds over any infinite field but here we need it only over $\mathbb{C}$). We state this lemma explicitly since we will also need it in our constructive proof.

Lemma 1. *Let $f_1, \dots, f_s \in \mathbb{Z}[x_1, \dots, x_n]$ be polynomials and V be the variety of $\mathbb{C}^n$ they define. Then for all coefficients $(\alpha_{i,j})_{i=1..s, j=1..n+1} \in \mathbb{C}^{s(n+1)}$ algebraically independent over $\mathbb{Q}$, the $n+1$ linear combinations $g_j = \sum_{i=1}^{s} \alpha_{i,j} f_i$ (for j from 1 to $n+1$) also define V.*

Unfortunately, in our case we cannot use transcendental numbers and must replace them by integers. The following lemma from [10] asserts that integers growing sufficiently fast will do. Once again, this is a weaker version adapted to our purpose.

Lemma 2. *Let $\phi(\alpha_1, \dots, \alpha_r)$ be a quantifier-free first-order formula over $(\mathbb{C}, 0, 1, +, -, \times, =)$, containing only polynomials of degree less than D and whose coefficients are integers of absolute value strictly bounded by C. Assume furthermore that $\phi(\bar{\alpha})$ holds for all coefficients $\bar{\alpha} = (\alpha_1, \dots, \alpha_r) \in \mathbb{C}^r$ algebraically independent over $\mathbb{Q}$.*

Then $\phi(\bar{\beta})$ holds for any sequence $(\beta_1, \dots, \beta_r)$ of integers satisfying $\beta_1 \geq C$ and $\beta_{j+1} \geq C D^j \beta_j^D$ (for $1 \leq j \leq r-1$).

The proof can be found in [10, Lemma 5.4] and relies on the lack of big integer roots of multivariate polynomials.

Let us sketch a first attempt to prove a constructive version of Lemma 1, namely that $n+1$ polynomials with integer coefficients are enough for defining V (this first try will not work but gives the idea of the proof of the next section). The idea is to use Lemma 2 with the formula $\phi(\bar{\alpha})$ that tells us that the $n+1$ linear combinations of the f_i with $\alpha_{i,j}$ as coefficients define the same variety as $f_1, \dots, f_s$. At first this formula is not quantifier-free, but over $\mathbb{C}$ we can eliminate quantifiers while keeping degree and coefficients reasonably small thanks to Theorem 1.

Lemma 1 asserts that $\phi(\bar{\alpha})$ holds as soon as the $\alpha_{i,j}$ are algebraically independent. Then Lemma 2 tells us that $\phi(\bar{\beta})$ holds for integers $\beta_{i,j}$ growing fast enough. Thus V is now defined by $n+1$ linear combinations of the f_i with integer coefficients.

In fact, this strategy fails to work for our purpose because the coefficients involved are growing too fast to be computed in polynomial space. That is why we will proceed by stages in the proofs below.

4.1 Tests of Membership

Lemma 3. *There exists a polynomial $q(n,d)$ such that, if $V \subseteq \mathbb{C}^n$ is a variety defined by $2(n+1)$ polynomials $f_1, \dots, f_{2(n+1)} \in \mathbb{Z}[x_1, \dots, x_n]$ of degree $\leq 2^d$ and of coefficients bounded by 2^{2^M} in absolute value, then:*

1. *the variety V is defined by $n+1$ polynomials $g_1, \dots, g_{n+1} \in \mathbb{Z}[x_1, \dots, x_n]$ of degree $\leq 2^d$ and of coefficients bounded by $2^{2^{M+q(n,d)}}$ in absolute value;*

2. *furthermore, the coefficients of the g_i are bitwise computable from those of the f_j in working space $Mq(n,d)$.*

Proof. The first-order formula $\phi(\bar{\alpha})$ (where $\bar{\alpha} \in \mathbb{C}^{2(n+1)^2}$), expressing that the $n+1$ linear combinations of the f_j's with coefficients $\bar{\alpha}$ also define V, can be written as follows:

$$\phi(\bar{\alpha}) \equiv \forall x \in \mathbb{C}^n \left(\bigwedge_{i=1}^{n+1} \sum_{j=1}^{2(n+1)} \alpha_{i,j} f_j(x) = 0 \leftrightarrow \bigwedge_{j=1}^{2(n+1)} f_j(x) = 0 \right),$$

where $\alpha_{i,j}$ is a shorthand for $\alpha_{2(i-1)(n+1)+j}$. The polynomials in this formula are of degree $\leq 1 + 2^d$ and their coefficients are bounded in absolute value by 2^{2^M}.

Over $\mathbb{C}$, the quantifier of this formula can be eliminated by Theorem 1: $\phi(\bar{\alpha})$ is equivalent to a quantifier-free formula $\psi(\bar{\alpha})$, the polynomials occuring in which have their degree less than $D = D(n, \log(3(n+1)), d+1)$ and their coefficients strictly bounded in absolute value by $C = 2^{2^{MD}}$, where $D(n, \log(3(n+1)), d+1) = 2^{p(n,\log(3(n+1)),d+1)}$ is defined in Theorem 1.

By Lemma 1, $\psi(\bar{\alpha})$ holds for all coefficients $\bar{\alpha}$ algebraically independent, so that we wish to apply Lemma 2 with integers β_i growing sufficiently fast. Let $r = (1 + 2(n+1)^2)p(n, \log(3(n+1)), d+1)$, so that

$$D \leq 2^r \text{ and } CD^{2(n+1)^2} \leq 2^{2^{M+r}}$$

and define

$$\beta_i = 2^{2^{M+2ir}} \text{ for } 1 \leq i \leq 2(n+1)^2.$$

Note that for all i, $\beta_i \leq 2^{2^{M+4(n+1)^2 r}}$, and it is furthermore easy to check that $\beta_1 \geq C$ and $\beta_{i+1} \geq CD^i \beta_i^D$. Thus by Lemma 2, $\psi(\bar{\beta})$ is true. Define the polynomial $q(n,d) = 1 + 4(n+1)^2 r$ (up to a multiplicative constant for the space complexity below). Now, letting

$$g_i = \sum_{j=1}^{2(n+1)} \beta_{i,j} f_j,$$

where $\beta_{i,j}$ is a shorthand for $\beta_{2(i-1)(n+1)+j}$, proves the first point of the theorem.

For the second point, remark that the coefficients β_i are bitwise computable in space $O(M + rn^2)$ and that the coefficients of the g_i are merely a sum of $2(n+1)$ products of β_j and coefficients of the f_k. This multiplication uses only space $O(M + rn^2)$ since the integers involved have encoding size $2^{O(M+rn^2)}$ (in our case this is particularly easy because the β_j are powers of 2). The $2n+1$ additions are also performed in space $O(M + rn^2)$. This proves the second point of the theorem. □

Proposition 4. *There exists a polynomial $p(n,s,d)$ such that, if V is a variety defined by 2^s polynomials $f_1, \ldots, f_{2^s} \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree $\leq 2^d$ and of coefficients bounded by 2^{2^M} in absolute value, then:*

1. *the variety V is defined by $n+1$ polynomials $g_1, \ldots, g_{n+1} \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree $\leq 2^d$ and of coefficients bounded by $2^{2^{M+p(n,s,d)}}$ in absolute value;*
2. *moreover, the coefficients of the g_i are bitwise computable from those of the f_j in working space $Mp(n,s,d)$.*

Proof. This is done by induction on s. Take $p(n,s,d) = sq(n,d)$ where $q(n,d)$ is the polynomial defined in Lemma 3. The base case $2^s \leq 2(n+1)$ follows from Lemma 3. Suppose therefore that $2^s > 2(n+1)$. Call V_1 and V_2 the varieties defined respectively by $f_1, \ldots, f_{2^{s-1}}$ and by $f_{2^{s-1}+1}, \ldots, f_{2^s}$.

Then $V = V_1 \cap V_2$ and by induction hypothesis, V_1 and V_2 are both defined by $n+1$ polynomials of degree $\leq 2^d$ whose coefficients are bounded by $2^{2^{M+(s-1)q(n,d)}}$ in absolute value and computable in space $M(s-1)q(n,d)$.

Therefore by Lemma 3, V is defined by $n+1$ polynomials of degree $\leq 2^d$ whose coefficients are bounded by $2^{2^{M+sq(n,d)}}$ in absolute value and computable in space $Msq(n,d)$ as claimed in the lemma. □

4.2 Union of Varieties

In our case, however, the tests made by the algorithm of Section 5 are not exactly of the form studied in the previous section: instead of a single variety given by s polynomials, we have to decide "$x \in W$?" when $W \subseteq \mathbb{C}^n$ is the union of k varieties. Of course, since the union is finite W is also a variety, but the encoding is not the same as above: now, k sets of s polynomials are given.

A first naive approach is to define $W = \cup_i V_i$ by the different products of the polynomials defining the V_i, but it turns out that there are too many products to be dealt with. Instead, we will adopt a divide-and-conquer scheme as previously.

Lemma 4. *There exists a polynomial $q(n,d)$ such that, if V_1 and V_2 are two varieties of $\mathbb{C}^n$, each defined by $n+1$ polynomials in $\mathbb{Z}[x_1,\dots,x_n]$, respectively $f_1,\dots,f_{n+1}$ and $g_1,\dots,g_{n+1}$, of degree $\leq 2^d$ and of coefficients bounded by 2^{2^M} in absolute value, then:*

1. *the variety $V = V_1 \cup V_2$ is defined by $n+1$ polynomials $h_1,\dots,h_{n+1}$ in $\mathbb{Z}[x_1,\dots,x_n]$ of degree $\leq 2^{d+1}$ and of coefficients bounded by $2^{2^{M+q(n,d)}}$ in absolute value;*
2. *the coefficients of the h_i are bitwise computable from those of the f_j and g_k in space $Mq(n,d)$.*

Proof. The variety V is defined by the $(n+1)^2$ polynomials $f_i g_j$ for $1 \leq i,j \leq n+1$: these polynomials have degree $\leq 2^{d+1}$. Note moreover that there are at most $2^{n(d+1)}$ monomials of fixed degree $\delta \leq 2^{d+1}$, therefore the coefficients of the $f_i g_j$ are a sum of at most $2^{n(d+1)}$ products of integers of encoding size 2^M. Thus they are computable in space $O(Mnd)$ from those of the f_i and g_j. This also shows that the coefficients of the products $f_i g_j$ are bounded in absolute value by $2^{n(d+1)}2^{2^{M+1}} \leq 2^{2^{M+1+n(d+1)}}$. Applying Proposition 4 now enables to conclude if we take $q(n,d) = 1 + n(d+1) + p(n, \log((n+1)^2), d+1)$, where p is the polynomial defined in Proposition 4. □

Proposition 5. *There exists a polynomial $r(n,s,k,d)$ such that, if $V_1,\dots,V_{2^k} \subseteq \mathbb{C}^n$ are 2^k varieties, V_i being defined by 2^s polynomials $f_1^{(i)},\dots,f_{2^s}^{(i)} \in \mathbb{Z}[x_1,\dots,x_n]$ of degree $\leq 2^d$ and of coefficients bounded by 2^{2^M} in absolute value, then:*

1. *the variety $V = \cup_{i=1}^{2^k} V_i$ is defined by $n+1$ polynomials $g_1,\dots,g_{n+1}$ in $\mathbb{Z}[x_1,\dots,x_n]$ of degree $\leq 2^{d+k}$ and whose coefficients are bounded in absolute value by $2^{2^{M+r(n,s,k,d)}}$;*
2. *moreover, the coefficients of the g_i are bitwise computable from those of the $f_{j'}^{(j)}$ in space $Mr(n,s,k,d)$.*

Proof. We proceed by induction on k. Define $r(n,s,k,d) = (k+1)(p(n,s,d+k) + q(n,d+k))$, where p and q are defined in Proposition 4 and Lemma 4 respectively. The base case $k=0$ is merely an application of Proposition 4. For $k > 0$, we first apply Proposition 4 to the V_i, so that each variety V_i is now defined by $n+1$ polynomials of degree $\leq 2^d$ and whose coefficients are bounded in absolute value by $2^{2^{M+p(n,s,d)}}$ and computable in space $Mp(n,s,d)$. Let us group the varieties V_i by pairs: call $W_i = V_{2i-1} \cup V_{2i}$ for $1 \leq i \leq 2^{k-1}$. There are 2^{k-1} varieties W_i and we have $V = \cup_i W_i$. By

Lemma 4, each variety W_i is defined by $n+1$ polynomials of degree $\leq 2^{d+1}$, of coefficients of bitsize $2^{M+p(n,s,d)+q(n,d)}$ and bitwise computable in space $M(p(n,s,d)+q(n,d))$. By induction hypothesis at rank $k-1$, V is defined by $n+1$ polynomials of degree $\leq 2^{d+1+(k-1)}$, of coefficients of bitsize $2^{M+p(n,s,d)+q(n,d)+k(p(n,\lceil\log(n+1)\rceil,d+k-1)+q(n,d+k-1))} \leq 2^{M+r(n,s,k,d)}$ and bitwise computable in space $Mr(n,s,k,d)$. This proves the lemma. □

Corollary 1. *Let $p(n)$ and $q(n)$ be two polynomials. Suppose $(f_n(\bar{x},\bar{y},\bar{z}))$ is a* uniform VPSPACE^0 *family with $|\bar{x}| = n$, $|\bar{y}| = p(n)$ and $|\bar{z}| = q(n)$. For an integer $0 \leq i < 2^{p(n)}$, call $V_i^{(n)} \subseteq \mathbb{C}^n$ the variety defined by the polynomials $f_n(\bar{x},i,j)$ for $0 \leq j < 2^{q(n)}$ (in this notation, i and j are encoded in binary).*

Then there exists a uniform VPSPACE^0 *family $g_n(\bar{x},\bar{y},\bar{z})$, where $|\bar{x}| = n$, $|\bar{y}| = p(n)$ and $|\bar{z}| = \lceil\log(n+1)\rceil$, such that*

$$\forall \bar{x} \in \mathbb{C}^n, \quad \forall k < 2^{p(n)}, \quad \left(\bar{x} \in \bigcup_{i=0}^{k} V_i^{(n)} \iff \bigwedge_{j=0}^{n} g_n(\bar{x},k,j) = 0\right).$$

Proof. If (f_n) is a uniform VPSPACE^0 family, by definition there exists a polynomial $p(n)$ such that the degree of f_n is bounded by $2^{p(n)}$ and the absolute value of the coefficients by $2^{2^{p(n)}}$. Therefore d, M, s and k are polynomially bounded in Proposition 5 and the space needed to compute the coefficients of g_n is polynomial. □

5 Proof of the Main Theorem

Sign conditions are the main ingredient of the proof. Over $\mathbb{C}$, we define the "sign" of $a \in \mathbb{C}$ by 0 if $a = 0$ and 1 otherwise. Let us fix a family of polynomials $f_1,\ldots,f_s \in \mathbb{Z}[x_1,\ldots,x_n]$. A *sign condition* is an element $S \in \{0,1\}^s$. Hence there are 2^s sign conditions. Intuitively, the i-th component of a sign condition determines the sign of the polynomial f_i.

5.1 Satisfiable Sign Conditions

The sign condition of a point $\bar{x} \in \mathbb{C}^n$ is the tuple $S^{\bar{x}} \in \{0,1\}^s$ defined by $S_i^{\bar{x}} = 0 \iff f_i(\bar{x}) = 0$. We say that a sign condition is *satisfiable* if it is the sign condition of some $\bar{x} \in \mathbb{C}^n$. As 0-1 tuples, sign conditions can be viewed as subsets of $\{1,\ldots,s\}$. Using a fast parallel sorting algorithm (e.g. Cole's, [6]), we can sort satisfiable sign conditions in polylogarithmic parallel time in a way compatible with set inclusion (e.g. the lexicographic order). We now fix such a compatible linear order on sign conditions and consider our satisfiable sign conditions $S^{(1)} < S^{(2)} < \ldots < S^{(N)}$ sorted accordingly.

The key point resides in the following theorem, coming from the algorithm of [8]: there is a "small" number of satisfiable sign conditions and enumerating them is "easy".

Theorem 3. *Let $f_1,\ldots,f_s \in \mathbb{Z}[x_1,\ldots,x_n]$ and d be their maximal degree. Then the number of satisfiable sign conditions is $N = (sd)^{O(n)}$, and there is a uniform algorithm working in space $(n\log(sd))^{O(1)}$ which, on boolean input $f_1,\ldots,f_s$ (in dense representation) and (i,j) in binary, returns the j-th component of the i-th satisfiable sign condition.*

When $\log(sd)$ is polynomial in n, as will be the case, this yields a PSPACE algorithm. If furthermore the coefficients of f_i are computable in polynomial space, we will then be able to use the satisfiable sign conditions in the coefficients of VPSPACE families, as in Lemma 5 below.

Let us explain why we are interested in sign conditions. An arithmetic circuit performs tests of the form $f(\bar{x}) = 0$ on input $\bar{x} \in \mathbb{C}^n$, where f is a polynomial. Suppose $f_1,\ldots,f_s$ is the list of all polynomials that can be tested in *any possible* computation. Then two elements of $\mathbb{C}^n$ with the same sign condition are simultaneously accepted or rejected by the circuit: the results of the tests are indeed always the same for both elements.

Thus, instead of finding out whether $\bar{x} \in \mathbb{C}^n$ is accepted by the circuit, it is enough to find out whether the sign condition of $\bar{x}$ is accepted. The advantage resides in handling only boolean tuples (the sign conditions) instead of complex numbers (the input $\bar{x}$). But we have to be able to find the sign condition of the input $\bar{x}$. This requires first the enumeration of all the polynomials possibly tested in any computation of the circuit.

5.2 Enumerating all Possibly Tested Polynomials

In the execution of an algebraic circuit, the values of some polynomials at the input $\bar{x}$ are tested to zero. In order to find the sign condition of the input $\bar{x}$, we have to be able to enumerate in polynomial space all the polynomials that can ever be tested to zero in the computations of an algebraic circuit. This is done as in [7, Th. 3] and [13].

Proposition 6. *Let C be a constant-free algebraic circuit with n variables and of depth d.*

1. *The number of different polynomials possibly tested to zero in the computations of C is $2^{d^2 O(n)}$.*
2. *There exists an algorithm using work space $(nd)^{O(1)}$ which, on input C and integers (i,j) in binary, outputs the j-th bit of the representation of the i-th polynomial.*

Proof. C is sliced in levels corresponding to the depth of the gates: input gates are on the level 0 and the output gate is the only one on level d.

Suppose that the results of the tests of the levels 0 to $i-1$ are fixed: we can then compute all the polynomials tested at level i. Since our algebraic circuits have fan-in at most 2, there are at most 2^{d-i} gates on level i of C: in particular, at most 2^{d-i} polynomials can be tested on level i. But the degree of a polynomial computed at level i is at most 2^i. Therefore, by Theorem 3 there are at most $(2^d)^{O(n)}$ possible outcomes for the tests of level i, and they are moreover enumerable in space $(nd)^{O(1)}$. Thus we can compute all the $(2^d)^{O(n)}$ possible outcomes of all the tests of level i and proceed inductively. This gives an algorithm using work space $(nd)^{O(1)}$ for enumerating all the polynomials that can possibly be tested in the executions of the circuit. Since there are $2^{dO(n)}$ possible outcomes at each level, the total number of polynomials for the whole circuit (that is, for d levels) is $(2^{dO(n)})^d = 2^{d^2 O(n)}$, as claimed in the statement of the theorem. □

Together with Theorem 3, this enables us to prove the following result which will be useful in the proof of Proposition 7: in uniform VPSPACE^0 we can enumerate the polynomials as well as the satisfiable sign conditions.

Lemma 5. *Let (C_n) be a uniform family of polynomial-depth algebraic circuits with polynomially many inputs. Call $d(n)$ the depth of C_n and $i(n)$ the number of inputs. Let $f_1^{(n)}, \dots, f_s^{(n)}$ be all the polynomials possibly tested to zero by C_n as in Proposition 6, where $s = 2^{O(nd(n)^2)}$. There are therefore $N = 2^{O(n^2 d(n)^2)}$ satisfiable sign conditions $S^{(1)}, \dots, S^{(N)}$ by Theorem 3.*

Then there exists a uniform VPSPACE^0 *family $(f_n(\bar{x}, \bar{y}, \bar{z}))$, where $|\bar{x}| = i(n)$, $|\bar{y}| = O(n^2 d(n)^2)$ and $|\bar{z}| = O(nd(n)^2)$, such that for all $1 \le i \le N$ and $1 \le j \le s$, we have:*

$$f_n(\bar{x}, i, j) = \begin{cases} 0 & \textit{if } S_j^{(i)} = 1 \\ f_j^{(n)}(\bar{x}) & \textit{otherwise.} \end{cases}$$

Proof. Computing the coefficients of the polynomials $f_1^{(n)}, \dots, f_s^{(n)}$ is done in polynomial space thanks to Proposition 6. Now, deciding whether $S_j^{(i)} = 1$ is also done in polynomial space thanks to Theorem 3. The lemma follows. □

5.3 Finding the Sign Condition of the Input

In order to find the sign condition $S^{\bar{x}}$ of the input $\bar{x} \in \mathbb{C}^n$, we will give a polynomial-time algorithm which tests some VPSPACE family for zero. Here is the formalized notion of a polynomial-time algorithm with VPSPACE tests.

Definition 3. *A polynomial-time algorithm with* uniform VPSPACE^0 *tests is a* uniform VPSPACE^0 *family $(f_n(x_1, \ldots, x_{u(n)}))$ together with a uniform family (C_n) of constant-free polynomial-size algebraic circuits endowed with special test gates of indegree $u(n)$, whose value is 1 on input $(a_1, \ldots, a_{u(n)})$ if $f_n(a_1, \ldots, a_{u(n)}) = 0$ and 0 otherwise.*

Observe that a constant number of uniform VPSPACE^0 families can be used in the preceding definition instead of only one: it is enough to combine them all in one by using "selection variables".

The precise result we show now is the following.

Proposition 7. *Let (C_n) be a uniform family of algebraic circuits of polynomial depth and with a polynomial number $i(n)$ of inputs. There exists a polynomial-time algorithm with* uniform VPSPACE^0 *tests which, on input $\bar{x} \in \mathbb{C}^{i(n)}$, returns the rank i of the sign condition $S^{(i)}$ of $\bar{x}$ with respect to the polynomials $g_1, \ldots, g_s$ tested to zero by C_n given by Proposition 6.*

Proof. Take the uniform VPSPACE^0 family $(f_n(\bar{x}, \bar{y}, \bar{z}))$ as in Lemma 5: in essence, f_n enumerates all the polynomials $g_1, \ldots, g_s$ possibly tested to zero in C_n and enumerates the N satisfiable sign conditions $S^{(1)} < \ldots < S^{(N)}$. The idea now is to perform a binary search in order to find the rank i of the sign condition of the input $\bar{x}$.

Let $S^{(j)} \in \{0,1\}^s$ be a satisfiable sign condition. We say that $S^{(j)}$ is a *candidate* whenever $\forall m \leq s$, $S^{(j)}_m = 0 \Rightarrow f_m(\bar{x}) = 0$. Remark that the sign condition of $\bar{x}$ is the smallest candidate. Call V_j the variety defined by the polynomials $\{g_m | S^{(j)}_m = 0\}$: by definition of f_n, V_j is also defined by the polynomials $f_n(\bar{x}, j, k)$ for $k = 1$ to s. Note that $S^{(j)}$ is a candidate if and only if $\bar{x} \in V_j$.

Lemma 5 combined with Corollary 1 assert that tests of the form $\bar{x} \in \cup_{k \leq j} V_k$ are in uniform VPSPACE^0. They are used to perform a binary search by making j vary. In a number of steps logarithmic in N (i.e. polynomial in n), we find the rank i of the sign condition of $\bar{x}$. □

5.4 A Polynomial-time Algorithm for PAR$_\mathbb{C}$ Problems

Lemma 6. *Let (C_n) be a uniform family of constant-free polynomial-depth algebraic circuits. There is a (boolean) algorithm using work space polynomial in n which, on input i, decides whether the elements of the i-th satisfiable sign condition $S^{(i)}$ are accepted by the circuit C_n.*

Proof. We follow the circuit C_n level by level. For test gates, we compute the polynomial f to be tested. Then we enumerate the polynomials $f_1, \ldots, f_s$ as in Proposition 6 for the circuit C_n and we find the index j of f in this list. By consulting the j-th bit of the i-th satisfiable sign condition with respect to $f_1, \ldots, f_s$ (which is done by the polynomial-space algorithm of Theorem 3), we therefore know the result of the test and can go on like this until the output gate. □

Theorem 4. *Let $A \in \mathrm{PAR}^0_\mathbb{C}$. There exists a polynomial-time algorithm with* uniform VPSPACE^0 *tests that decides A.*

Proof. A is decided by a uniform family (C_n) of constant-free polynomial-depth algebraic circuits. On input $\bar{x}$, thanks to Proposition 7 we first find the rank i of the sign condition of $\bar{x}$ with respect to the polynomials $f_1, \ldots, f_s$ of Proposition 6. Then we conclude by a last uniform VPSPACE^0 test simulating the polynomial-space algorithm of Lemma 6 on input i. □

Theorem 2 follows immediately from this result. One could obtain other versions of these two results by changing the uniformity conditions or the role of constants.

References

1. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer-Verlag, 1998.
2. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.
3. P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics.* Springer, 2000.
4. O. Chapuis and P. Koiran. Saturation and stability in the theory of computation over the reals. *Annals of Pure and Applied Logic*, 99:1–49, 1999.
5. P. Charbit, E. Jeandel, P. Koiran, S. Perifel, and S. Thomassé. Finding a vector orthogonal to roughly half a collection of vectors. Available from http://perso.ens-lyon.fr/pascal.koiran/publications.html. Accepted for publication in *Journal of Complexity*, 2006.
6. R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.
7. F. Cucker and D. Grigoriev. On the power of real Turing machines over binary inputs. *SIAM Journal on Computing*, 26(1):243–254, 1997.
8. N. Fitchas, A. Galligo, and J. Morgenstern. Precise sequential and parallel complexity bounds for quantifier elimination over algebraically closed fields. *Journal of Pure and Applied Algebra*, 67:1–14, 1990.
9. D. Grigoriev. Topological complexity of the range searching. *Journal of Complexity*, 16:50–53, 2000.
10. P. Koiran. Randomized and deterministic algorithms for the dimension of algebraic varieties. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pages 36–45, 1997.
11. P. Koiran. Circuits versus trees in algebraic complexity. In *Proc. STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 35–52. Springer-Verlag, 2000.
12. P. Koiran and S. Perifel. Valiant's model: from exponential sums to exponential products. In *Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 596–607. Springer-Verlag, 2006.
13. P. Koiran and S. Perifel. VPSPACE and a transfer theorem over the reals. Available from http://perso.ens-lyon.fr/pascal.koiran/publications.html, 2006.
14. G. Malod. *Polynômes et coefficients.* PhD thesis, Université Claude Bernard Lyon 1, July 2003. Available from http://tel.archives-ouvertes.fr/tel-00087399.
15. G. Malod and N. Portier. Characterizing Valiant's algebraic complexity classes. In *Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 704–716. Springer-Verlag, 2006.
16. B. Poizat. *Les petits cailloux.* Aléas, 1995.
17. L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.