VPSPACE and a Transfer Theorem over the Reals

Pascal Koiran, Sylvain Perifel

VPSPACE and a Transfer Theorem over the Reals

Pascal Koiran and Sylvain Perifel

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

Abstract We introduce a new class VPSPACE of families of polynomials. Roughly speaking, a family of polynomials is in VPSPACE if its coefficients can be computed in polynomial space. Our main theorem is 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 $\mathrm{P}_{\mathbb{R}}$. As a result, one must first be able to show that there are VPSPACE families which are hard to evaluate in order to separate $\mathrm{P}_{\mathbb{R}}$ from $\mathrm{NP}_{\mathbb{R}}$, or even from $\mathrm{PAR}_{\mathbb{R}}$.
Keywords: computational complexity, algebraic complexity, Blum-Shub-Smale model, Valiant's model.

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 [19]. Deciding whether a multivariate polynomial has a real root is a typical example of a decision problem. This problem is NP-complete in the Blum-Shub-Smale model of computation over the real numbers [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]. In that paper, we showed that if certain polynomials can be evaluated efficiently then certain decision problems become easy. The polynomials considered in [12] are those that can be written as exponential-size products of polynomials that are easy to compute (see [12] for a precise definition) over some field K. The decision problems under consideration are those that are in NP in the structure $(K, +, -, =)$, in which multiplication is not allowed.

In the present paper we work with a larger class of polynomial families, which we call VPSPACE. Roughly speaking, a family of polynomials (of possibly exponential degree) is in VPSPACE if its coefficients can be evaluated in

* UMR 5668 ENS Lyon, CNRS, UCBL, INRIA. Research report RR2006-29.

polynomial space. For instance, we show that resultants of systems of multivariate polynomial equations form a VPSPACE family. Our main result is 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 $\mathrm{P}_{\mathbb{R}}$. This result relies crucially on a combinatorial lemma due to Grigoriev [10] and especially on its effective version, recently established in [7]. The class $\mathrm{PAR}_{\mathbb{R}}$ plays roughly the same role in the theory of computation over the reals as PSPACE in discrete complexity theory. In particular, it contains $\mathrm{NP}_{\mathbb{R}}$ [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 $\mathrm{P}_{\mathbb{R}}$ from $\mathrm{NP}_{\mathbb{R}}$, or even from $\mathrm{PAR}_{\mathbb{R}}$, 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 notions and notations from algebraic complexity (Valiant's model, the Blum-Shub-Smale model). A uniform version of the class VPSPACE is defined in Section 3. The next two sections of the paper are devoted to the transfer theorem. Section 4 deals with sign conditions, an important tool from computational real algebraic geometry. The transfer theorem is stated at the beginning of Section 5, and proved thereafter. Some of the proofs are relegated to the appendix. In addition, the appendix contains several other results. In particular, we show in Appendix A that resultants of multivariate polynomial systems form a VPSPACE family. The definition of the nonuniform class VPSPACE is given in Appendix B, and the hypothesis that VPSPACE families are easy to evaluate is discussed in Appendix C. We show that (assuming the generalized Riemann hypothesis) this hypothesis is equivalent to: VP = VNP and P/poly = PSPACE/poly. The conjunction of these two equalities is an extremely strong assumption: by results from [3] (see [11]), it implies, assuming again GRH, that NC/poly = PSPACE/poly. This conjunction of equalities is still apparently consistent with our current understanding of complexity theory. We also discuss the uniform, constant-free version of the hypothesis that VPSPACE families are easy to evaluate. It turns out that this stronger hypothesis implies that PSPACE collapses to the polynomial-time uniform version of NC. Such a dramatic collapse of complexity classes looks extremely unlikely, but as far as we know it cannot be refuted with the current methods of complexity theory.

2 Preliminaries

The notions of boolean complexity theory that we use are quite standard. In the present section, we focus on algebraic complexity.

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 ordered field $(\mathbb{R}, +, -, \times, \leq)$ of

the real numbers. Although the original definitions of Blum, Shub and Smale [2,1] are in terms of uniform machines, we will follow [17] by using families of algebraic circuits to recognize languages over $\mathbb{R}$, that is, subsets of $\mathbb{R}^\infty = \bigcup_{n\geq 0} \mathbb{R}^n$.

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 $\mathbb{R}^n$ to $\mathbb{R}$. On input $\bar{u} \in \mathbb{R}^n$ 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 > 0 and 1 otherwise. We assume without loss of generality that the output is a test gate. The value returned by the circuit is therefore 0 or 1.

The class $\mathrm{P}_\mathbb{R}$ is the set of languages $L \subseteq \mathbb{R}^\infty$ such that there exists a tuple $\bar{a} \in \mathbb{R}^p$ 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 $\bar{x} \in \mathbb{R}^n$, $\bar{x} \in L \Leftrightarrow C_n(\bar{x}, \bar{a}) = 1$. 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 [6], we define the class $\mathrm{PAR}_\mathbb{R}$ as the set of languages over $\mathbb{R}$ recognized by a PSPACE-uniform family of algebraic circuits of polynomial depth (and possibly exponential size), with constants $\bar{a}$ as for $\mathrm{P}_\mathbb{R}$. 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 $\mathrm{P}^0_\mathbb{R}$ and $\mathrm{PAR}^0_\mathbb{R}$.

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 $\mathbb{R}$; 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, $f_n \in \mathbb{R}[x_1, \ldots, x_{u(n)}]$. The class $\mathrm{VP}_{\mathrm{nb}}$ 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 $\mathrm{VP}_{\mathrm{nb}}$ 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 $\mathrm{VP}^0_{\mathrm{nb}}$ (for bounded-degree classes, we are to be more careful; see Definition 6).

Note at last that arithmetic circuits are at least as powerful as boolean circuits in the sense that one can simulate the latter by the former. Indeed, we can for instance replace $\neg u$ by $1 - u$, $u \wedge v$ by uv, and $u \vee v$ by $u + v - uv$. This proves the following classical lemma.

Lemma 1. *Any boolean circuit C can be simulated by an arithmetic one of size at most $3|C|$, in the sense that on boolean inputs, both circuits output the same value.*

3 The Class VPSPACE

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 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 define first uniform 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 defined in the appendix is nonuniform and allows for arbitrary constants.

3.2 An Alternative Characterization

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 VPSPACE^0.

Proposition 1. *The two classes* uniform VPSPACE^0 *and* uniform VPAR^0 *are equal.*

Proof. Let (f_n) be a uniform VPSPACE^0 family. In order to compute f_n by an arithmetic circuit of polynomial depth, we compute all its monomials in parallel and sum them in a divide-and-conquer-fashion. The resulting family of arithmetic circuits is uniform due to the uniformity condition on (f_n).

For the converse, take an arithmetic circuit of polynomial depth. We show that we can build a boolean circuit of polynomial depth which takes as input the encoding α of a monomial and computes the coefficient of $\bar{x}^\alpha$. We proceed by induction, computing the coefficient of $\bar{x}^\alpha$ for each gate of the original arithmetic circuit. For the input gates, this is easy. For a $+$-gate, it is enough to add both coefficients. For a gate $a \times b$, we compute in parallel the sum of the cd over all the monomials $\bar{x}^\beta$ and $\bar{x}^\gamma$ such that $\beta + \gamma = \alpha$, where c is the coefficient of $\bar{x}^\gamma$ in the gate a, and d the coefficient of $\bar{x}^\beta$ in the gate b. The whole boolean circuit remains uniform and of polynomial depth. Therefore, the coefficient function is in PSPACE by the "parallel computation thesis". □

We see here the similarity with $\mathrm{PAR}_{\mathbb{R}}$, 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 VPSPACE^0.

4 Sign Conditions

4.1 Definition

Given are s polynomials $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$. A sign condition is merely an s-tuple $S \in \{-1, 0, 1\}^s$. Intuitively, the i-th coordinate of S represents the

sign of f_i: -1 for < 0, 0 for 0, and 1 for > 0. Accordingly, the sign condition of a point $\bar{x} \in \mathbb{R}^n$ is the tuple $S \in \{-1, 0, 1\}^s$ such that $S_i = -1$ if $f_i(\bar{x}) < 0$, $S_i = 0$ if $f_i(\bar{x}) = 0$ and $S_i = 1$ if $f_i(\bar{x}) > 0$.

Of course some sign conditions are not realizable, in the sense that the polynomials can nowhere take the corresponding signs (think for instance of $x^2 + 1$ which can only take positive values over $\mathbb{R}$). We say that a sign condition is *satisfiable* if it is the sign condition of some $\bar{x} \in \mathbb{R}^n$ and we call N the number of satisfiable sign conditions. The key result detailed in the next section is that among all possible sign conditions, there are few satisfiable ones (i.e. N is small), and there exists a polynomial space algorithm to enumerate them all.

4.2 A PSPACE Algorithm for Sign Conditions

The following theorem of Renegar [18, Prop. 4.1] will prove to be a central tool in our proofs.

Theorem 1 (Renegar). *Let $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$ be s polynomials of maximal degree d, and whose coefficients have bit size $\leq L$. Then:*

1. *there are $N = (sd)^{O(n)}$ satisfiable sign conditions;*
2. *there is an algorithm using work space $(\log L)[n \log(sd)]^{O(1)}$ which, on input $(f_1, \ldots, f_s)$ in dense representation, and (i, j) in binary, outputs the j-th component of the i-th satisfiable sign condition.*

Note that if $d = 2^{n^{O(1)}}$, $s = 2^{n^{O(1)}}$ and $L = 2^{n^{O(1)}}$ as will be the case, then the work space of the algorithm is polynomial in n.

4.3 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. If two points $\bar{x}$ and $\bar{y}$ have the same sign condition with respect to all polynomials possibly tested to zero, then they will either both belong to the language, or both be outside of it: indeed the results of all the tests will be the same during the execution of the circuit. Therefore we can handle sign conditions (i.e. boolean words) instead of algebraic inputs.

Note that 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 some computation of an algebraic circuit. This is done as in [9, Th. 3].

Proposition 2. *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 some computation 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 i-th of these polynomials.*

The proof can be found in Appendix E. Note that this proposition can also be useful when our algebraic circuit is not constant-free: it is enough to replace the constants by fresh variables. The only risk is indeed to take more polynomials into account since we have replaced specific constants by generic variables.

5 A Transfer Theorem

In this section we prove our main result.

Theorem 2. Uniform $\mathrm{VPSPACE}^0$ = uniform $\mathrm{VP}^0_{\mathrm{nb}} \Longrightarrow \mathrm{PAR}^0_{\mathbb{R}} = \mathrm{P}^0_{\mathbb{R}}$.

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

Let $A \in \mathrm{PAR}^0_{\mathbb{R}}$: it is decided by a uniform family (C_n) of constant-free algebraic circuits of polynomial depth. For convenience, we fix n and work with C_n. For the proof of Theorem 2 we will need to find the sign condition of the input $\bar{x}$ with respect to the polynomials $f_1, \ldots, f_s$ of Proposition 2, that is to say, with respect to all the polynomials that can be tested to zero in an execution of C_n. We denote by N the number of satisfiable sign conditions with respect to $f_1, \ldots, f_s$.

Note that most of the forthcoming results depend on the polynomials $f_1, \ldots, f_s$, therefore on the choice of C_n. For instance, once C_n and $f_1, \ldots, f_s$ are chosen, the satisfiable sign conditions are fixed and we will speak of the i-th satisfiable sign condition without referring explicitly to the polynomials $f_1, \ldots, f_s$.

In order to find the sign condition of the input, 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 $\mathrm{VPSPACE}^0$ *tests is a* uniform $\mathrm{VPSPACE}^0$ *family* $(f_n(x_1, \ldots, x_{u(n)}))$ *together with a uniform constant-free family* (C_n) *of 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* $f_n(a_1, \ldots, a_{u(n)}) \leq 0$ *and* 0 *otherwise.*

Observe that a constant number of uniform $\mathrm{VPSPACE}^0$ 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 following Theorem 3 is the main result en route to showing the transfer theorem. It is proved via successive lemmas in Sections 5.1 to 5.3: we proceed as in [10] but constructively.

Theorem 3. *There is a polynomial-time algorithm with* uniform $\mathrm{VPSPACE}^0$ *tests that, on input* $\bar{x}$*, computes the rank of the sign condition of* $\bar{x}$ *with respect to* $f_1, \ldots, f_s$*.*

5.1 Truncated Sign Conditions

A truncated sign condition is merely an element T of $\{0, 1\}^s$. Contrary to full sign conditions, only the two cases $= 0$ and $\neq 0$ are distinguished. We define in a natural way the truncated sign condition T of a point $\bar{x}$: $T_i = 0$ if and only if $f_i(\bar{x}) = 0$.

Of course, there are fewer satisfiable truncated sign conditions than full ones, and of course there exists a polynomial space algorithm to enumerate them. Furthermore, truncated sign conditions can be viewed as subsets of $\{1, \ldots, s\}$ (via the convention $k \in T \iff T_k = 1$), therefore enabling us to speak of inclusion of truncated sign conditions.

We fix an order $\leq_T$ compatible with inclusion and easily computable in parallel, e.g. the lexicographic order. Let us call $T^{(i)}$ the i-th satisfiable truncated sign condition with respect to this order.

Lemma 2. *There is an algorithm using work space polynomial in n which, on input $(f_1, \ldots, f_s)$ in dense representation, and (i, j) in binary, outputs the j-th component of $T^{(i)}$ (the i-th satisfiable truncated sign condition with respect to $\leq_T$).*

Proof. It is enough to use the algorithm of Theorem 1, followed by a fast parallel-sorting procedure, for instance Cole's parallel merge-sort algorithm [8]. □

Note that the truncated sign condition of the input $\bar{x}$ is the maximal truncated satisfiable sign condition T satisfying $\forall i, T_i = 1 \Rightarrow f_i(\bar{x}) \neq 0$. Hence we are to find a maximum. This will be done by binary search.

Lemma 3. *There is a* uniform VPSPACE^0 *family (g_n) of polynomials satisfying, for real $\bar{x}$ and boolean i,*

$$g_n(\bar{x}, i) = \prod_{j \leq i} \Big(\sum_{k \notin T^{(j)}} f_k(\bar{x})^2 \Big).$$

Proof. Lemma 2 asserts that deciding whether $k \notin T^{(j)}$ is in PSPACE. Then we use twice Lemma 13 from Appendix D (once for the sum and once for the product). □

Proposition 3. *There is a polynomial-time algorithm with* uniform VPSPACE^0 *tests which on input $\bar{x}$ outputs the rank m of its truncated sign condition $T^{(m)}$.*

Proof. The algorithm merely consists in performing a binary search thanks to the polynomials of Lemma 3: if the truncated sign condition of the input $\bar{x}$ is $T^{(m)}$, then $\prod_{j \leq i} \big(\sum_{k \notin T^{(j)}} f_k(\bar{x})^2 \big) = 0$ if and only if $m \leq i$. By making i vary, we find m in a number of steps logarithmic in the number of satisfiable truncated sign conditions, i.e. in polynomial time. □

5.2 Binary Search for the Full Sign Condition

We say that a (full) sign condition S is compatible with the truncated sign condition T if $\forall i, T_i = 0 \Leftrightarrow S_i = 0$ (i.e. they agree for "$= 0$" and for "$\neq 0$"). Let N' denote the number of (full) satisfiable sign conditions compatible with the truncated sign condition of the input $\bar{x}$. Obviously, $N' \leq N$. The following lemma is straightforward after Lemma 2 and Theorem 1.

Lemma 4. *There is an algorithm using work space polynomial in n which, on input (i, j, k), ouputs the j-th bit of the i-th satisfiable sign condition compatible with $T^{(k)}$.*

Since we know the truncated sign condition of $\bar{x}$ after running the algorithm of Proposition 3, we know which polynomials vanish at $\bar{x}$. We can therefore discard the zeros in the (full) compatible satisfiable sign conditions. Hence we are now concerned with two-valued sign conditions, that is, elements of $\{-1, 1\}^{s'}$ with $s' \leq s$. In what follows arithmetic over the field of two elements will be used, hence it will be simpler to consider that our sign conditions have value among $\{0, 1\}$ instead of $\{-1, 1\}$: 0 for > 0 and 1 for < 0. Thus sign conditions are viewed as vectors over $\{0, 1\}$, or alternately as subsets of $\{1, \ldots, s'\}$. The set $\{0, 1\}^{s'}$ is endowed with the inner product $u.v = \sum_i u_i v_i (\mathrm{mod}\ 2)$, and we say that u and v are orthogonal whenever $u.v = 0$ (see [7]).

The following proposition from [7] will be useful. It consists in an improvement of the result of [10]: first (and most importantly), it is constructive, and second, the range $[N'/2 - \sqrt{N'}/2, N'/2 + \sqrt{N'}/2]$ here is much better than the original one $[N'/3, 2N'/3]$.

Proposition 4. *Let V be a set of N' vectors of $\{0, 1\}^{s'}$.*

1. *There exists a vector u orthogonal to at least $N'/2 - \sqrt{N'}/2$ and at most $N'/2 + \sqrt{N'}/2$ vectors of V.*
2. *Such a vector u can be found on input V by a logarithmic space algorithm.*

Our aim is to find the sign condition of $\bar{x}$. We will use Proposition 4 in order to divide the cardinality of the search space by two at each step. This is based on the following observation: if $u \in \{0, 1\}^{s'}$, the value of the product $\prod_{j \in u} f_j(\bar{x})$ is negative if the inner product of u and the sign condition of $\bar{x}$ is 1, and is positive otherwise. The idea is then to choose u judiciously so that the number of satisfiable sign conditions having the same inner product with u as the sign condition of $\bar{x}$ is halved at each step. Therefore, in a logarithmic number of steps, the sign condition of $\bar{x}$ will be uniquely determined. This gives the following algorithm for finding the sign condition of $\bar{x}$.

- Let E be the set of all the satisfiable sign conditions.
- While E contains more than one element, do
  - Find by Proposition 4 a vector u orthogonal to at least $|E|/2 - \sqrt{|E|}/2$ and at most $|E|/2 + \sqrt{|E|}/2$ vectors of E.
  - Let b be the result of the test "$\prod_{j \in u} f_j(\bar{x}) < 0$?".
  - Let the new E be the set of all sign conditions in E which have inner product b with u.
- Enumerate all the satisfiable sign conditions and find the one that produces exactly the same results as in the loop: this is the sign condition of $\bar{x}$.

Note that the number of steps is $O(\log N')$, which is polynomial in n. The last step of this algorithm (namely, recovering the rank of the sign condition of $\bar{x}$ from the list of results of the loop) is detailed in Section 5.3.

We now show how to perform this algorithm in polynomial time with uniform VPSPACE^0 tests. The main technical difficulty is that according to Definition 3 we can use only one VPSPACE family, whereas we want to make adaptative tests. We therefore have to store the intermediate results of the preceding tests in some variables $\bar{c}$ (a "list of choices") of the VPSPACE polynomial. Proposition 4 shows that, by reusing space, there exists a logspace algorithm that, given any set V of N' vectors together with a "list of choices" $c \in \{0,1\}^l$ (with $l = O(\log N')$), enumerates $l+1$ vectors $u^{(1)}, \ldots, u^{(l+1)}$ satisfying the following condition ($\star$):

- $u^{(1)}$ is orthogonal to at least $N'/2 - \sqrt{N'}/2$ and at most $N'/2 + \sqrt{N'}/2$ vectors of V.
- Let $V_i \subseteq V$ be the subset of all the vectors $v \in V$ satisfying $\forall j \leq i, v.u^{(j)} = c_j$. Then the vector $u^{(i+1)}$ is orthogonal to at least $|V_i|/2 - \sqrt{|V_i|}/2$ and at most $|V_i|/2 + \sqrt{|V_i|}/2$ vectors of V_i.

Note that $|V_i|$ is roughly divided by 2 at each step, so the number of steps is $O(\log N')$. In particular, since s' and N' are simply exponential, the following lemma is easily derived by combining what precedes with Lemma 4.

Lemma 5. *There is an algorithm using work space polynomial in n which, on input (i,j,k,c) in binary, outputs the j-th bit of $u^{(i)} \in \{0,1\}^{N'}$, where the vectors $u^{(1)}, \ldots, u^{(l+1)}$ satisfy condition ($\star$) for the input consisting of:*

- *the set V of the N' (full) satisfiable sign conditions compatible with $T^{(k)}$,*
- *together with the list of choices $c \in \{0,1\}^l$.*

Lemma 6. *There exists a* uniform VPSPACE^0 *family (h_n) satifsying, for real $\bar{x}$ and boolean (i,k,c):*

$$h_n(\bar{x}, i, k, c) = \prod_{j \in u^{(i)}} f_j(\bar{x}),$$

where $u^{(1)}, \ldots, u^{(l+1)}$ are defined as in Lemma 5 (in particular they depend on $T^{(k)}$).

Proof. Lemma 5 asserts that deciding whether $j \in u^{(i)}$ is done in polynomial space. The use of Lemma 13 from Appendix D then concludes the proof. □

Therefore, by a uniform VPSPACE^0 test, one is able to know the sign of the polynomial $h_n(\bar{x}, i, k, c) = \prod_{j \in u^{(i)}} f_j(\bar{x})$. As mentioned before, this gives us the inner product of $u^{(i)}$ and the (full) sign condition of $\bar{x}$: this sign is < 0 if and only if the inner product is 1. By beginning with $c = 0 \cdots 0$ (step 1), and at step $i \geq 2$ letting $c_{i-1} = 1$ if and only if the preceding test was < 0, the number of sign conditions that have the same inner products as that of $\bar{x}$ is divided by (roughly) two at each step. At the end, we therefore have a list of choices c that only the sign condition of $\bar{x}$ fulfills. This proves the following lemma.

Lemma 7. *There is a polynomial-time algorithm with* uniform $\mathrm{VPSPACE}^0$ *tests which on input $\bar{x}$ outputs the list of choices c (defined as above) which uniquely characterizes the sign condition of $\bar{x}$, provided we know the rank k of the truncated sign condition $T^{(k)}$ of $\bar{x}$.*

We are now able to recover the rank of the sign condition of $\bar{x}$ from this information, as explained in the next section.

5.3 Recovering the Rank of the Sign Condition

Lemma 8. *There is an algorithm using work space polynomial in n which, on input $c \in \{0,1\}^l$ (a list of choices) and k, outputs the rank of a satisfiable sign condition compatible with $T^{(k)}$ that fulfills the list of choices c.*

Proof. In polynomial space we recompute all the vectors $u^{(i)}$ as in Lemma 5, then we enumerate all the sign conditions thanks to Theorem 1 until we find one that fulfills the list of choices c. □

The proof of Theorem 3 now follows easily from Proposition 3 and Lemmas 7 and 8.

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

Remember that $A \in \mathrm{PAR}^0_{\mathbb{R}}$ and (C_n) is a uniform family of polynomial-depth algebraic circuits deciding A.

Lemma 9. *There is a (boolean) algorithm using work space polynomial in n which, on input i (the rank of a satisfiable sign condition), decides whether the elements of the sign condition S 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 2 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 1), we therefore know the result of the test and can go on like this until the output gate. □

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

Proof. A is decided by a uniform family (C_n) of polynomial depth algebraic circuits. On input $\bar{x}$, thanks to Theorem 3 we first find the rank of the sign condition of $\bar{x}$ with respect to the polynomials $f_1, \ldots, f_s$ of Proposition 2. Then we conclude by Lemma 9. □

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*. Number 7 in Algorithms and Computation in Mathematics. Springer, 2000.
4. P. Bürgisser and M. Lotz. The complexity of computing the Hilbert polynomial of smooth equidimensional complex projective varieties. Technical report, University of Paderborn, 2005. arXiv:cs.SC/0502044 v 1, 8 Feb 2005.
5. J. F. Canny. Generalized characteristic polynomials. In *Proc. ISSAC'88*, pages 293–299, 1988.
6. 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.
7. 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.
8. R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.
9. F. Cucker and D. Grigoriev. On the power of real Turing machines over binary inputs. *SIAM Journal on Computing*, 26(1):243–254, 1997.
10. D. Grigoriev. Topological complexity of the range searching. *Journal of Complexity*, 16:50–53, 2000.
11. P. Koiran. Valiant's model and the cost of computing integers. *Computational Complexity*, 13:131–146, 2004.
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. F. S. Macaulay. Algebraic theory of modular systems. *Cambridge tracts*, 19, 1916.
14. G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard Lyon 1, July 2003.
15. G. L. Miller, V. Ramachandran, and E. Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM J. Computing*, 17(4):687–695, 1988.
16. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
17. B. Poizat. *Les petits cailloux*. Aléas, 1995.
18. J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, part 1. *Journal of Symbolic Computation*, 13:255–299, 1992.
19. L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.
20. L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.

A An Example

Algebraic geometry is a natural source of examples for the study of polynomials from a computational point of view. For instance, the Hilbert polynomial is studied in [4] from the point of view of discrete complexity theory. Here we study a different example: the computation of the resultant of a system of multivariate polynomials. A system of $n+1$ homogenous equations in $n+1$ complex variables has a nontrivial solution if and only if its resultant is zero. We sketch the construction of the resultant below. More details can be found for instance in [13] or [5].

Let $f_1, \ldots, f_{n+1} \in \mathbb{C}[X_0, \ldots, X_n]$ be a system of $n+1$ homogeneous polynomials. The resultant consists in the quotient of the determinants of two matrices M and M':

$$R = \frac{\det M}{\det M'} \tag{1}$$

where the coefficients of M are among those of the f_i's, and M' is a submatrix of M. The matrix M is called Macaulay's matrix (a generalization of Sylvester's for two univariate polynomials) and is described as follows. Let d_i be the degree of f_i and $d = 1 + \sum_{i=1}^{n+1}(d_i - 1)$. Denote by Mon_d the set of all monomials in $X_0, \ldots, X_n$ of degree d: the cardinal of Mon_d is $N = \binom{d+n}{d}$.

The matrix M has N rows and N columns, both indexed by the elements of Mon_d. The row corresponding to the monomial $\bar{x}^\alpha$ represents the polynomial

$$\frac{\bar{x}^\alpha}{x_i^{d_i}} f_i, \text{ where } i = \min\{j; x_j^{d_j} \text{ divides } \bar{x}^\alpha\}.$$

Finally, the submatrix M' consists in the rows and columns of M that are "not reduced", see [5]. What we will compute is not the resultant R itself but rather a multiple of it, namely $\det M$. Whenever $\det M' \neq 0$, this does not change anything if we are only concerned by the vanishing of R.

From now on, we will assume for simplicity that all the d_i are equal. We will let n go to infinity, but the common value δ of the d_i will remain constant. A system $(f_1, \ldots, f_{n+1})$ of $n+1$ homogeneous polynomials of degree δ in $n+1$ variables is encoded by the list of the coefficients of the polynomials, i.e., by $k(n+1)$ variables $(a_{1,1}, \ldots, a_{1,k}, a_{2,1}, \ldots, a_{n+1,k})$ where $k = \binom{n+\delta}{\delta}$ is the number of monomials of degree δ in $n+1$ variables. Note that k is polynomial in n for any fixed δ.

The matrix $\mathrm{Mac}_n^\delta(f_1, \ldots, f_{n+1})$ is then defined as the Macaulay matrix M of $(f_1, \ldots, f_{n+1})$. This matrix is of size $\binom{n+d}{d}$, where $d = 1 + (n+1)(\delta - 1)$. This is exponential in n as soon as $\delta \geq 2$. Computing the determinant of M can be done by a circuit of depth polylogarithmic in the size of M, thus polynomial in n. The above considerations then prove the following proposition.

Proposition 5. *For any fixed δ, the family* $(\det(\mathrm{Mac}_n^\delta))$ *(the determinant of the Macaulay matrix of a system of $n+1$ homogeneous polynomials of degree δ in $n+1$ variables) is in* uniform $\mathrm{VPSPACE}^0$.

Likewise, the determinants of the matrices M' in (1) form a uniform VPSPACE^0 family.

B The Non-Uniform Class VPSPACE

We begin with the definitions of the nonuniform classes VPSPACE^0 and VPSPACE. Note that the only difference between VPSPACE^0 and uniform VPSPACE^0 is the nonuniformity of the coefficient function.

Definition 4. *The class* 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/poly*.*

Now, the class VPSPACE *is the set of all families* $(f_n(\bar{x}))$ *of multivariate polynomials* $f_n \in K[x_1, \ldots, x_{u(n)}]$ *such that there exist a family* $(g_n(\bar{x}, \bar{y})) \in$ VPSPACE^0 *together with a family of tuples of constants* $(\bar{a}^{(n)})$ *satisfying for all* n*:*

$$f_n(\bar{x}) = g_n(\bar{x}, \bar{a}^{(n)}).$$

We introduce temporarily a degree-bounded version of VPSPACE: this will prove useful for comparing VPSPACE to VP and VNP since the degree of the polynomials in these last two classes are polynomially bounded. A family (f_n) of polynomials is in VPSPACE$^0_{\mathrm{b}}$ if $(f_n) \in$ VPSPACE^0 and the size of the coefficients as well as the degree of f_n are polynomially bounded. The class VPSPACE$_{\mathrm{b}}$ is then defined from VPSPACE$^0_{\mathrm{b}}$ in the same way as VPSPACE is defined from VPSPACE^0 in Definition 4. This new class is interesting for our purpose due to the following two lemmas.

Lemma 10.

$$\mathrm{VPSPACE}_{\mathrm{b}} = \mathrm{VP} \iff \mathrm{VPSPACE} = \mathrm{VP}_{\mathrm{nb}}.$$

Proof. Assume first that VPSPACE $=$ VP$_{\mathrm{nb}}$, and take a family $(f_n) \in$ VPSPACE$_{\mathrm{b}}$. Since VPSPACE$_{\mathrm{b}} \subset$ VPSPACE, (f_n) is in fact in VP$_{\mathrm{nb}}$ by hypothesis. Now, since the degree of (f_n) is polynomially bounded, $(f_n) \in$ VP.

For the converse, take a family $(f_n) \in$ VPSPACE: remember that it can be written as $f_n(\bar{x}) = g_n(\bar{x}, \bar{a}^{(n)})$ for some constants $\bar{a}^{(n)}$ and $(g_n(\bar{x}, \bar{y})) \in$ VPSPACE^0. For convenience, let us rename the $u(n)$ variables of g_n by $v_1, \ldots, v_{u(n)}$, thus we have:

$$g_n(\bar{v}) = \sum_{\alpha} \Big((-1)^{a(n,\alpha,0)} \sum_{i=1}^{2^{p(n)}} a(n,\alpha,i) 2^{i-1} \bar{v}^{\alpha} \Big),$$

where a is in PSPACE/poly. In this expression, $p(n)$ is a polynomial and $2^{p(n)}$ bounds the size of the coefficients as well as the degree of g_n. In order to use the hypothesis, we have to somehow define a family $(h_n) \in \mathrm{VPSPACE}^0_{\mathrm{b}}$ that will "simulate" (g_n). Let us define

$$\Big(h_n(z_{1,1},\ldots,z_{1,p(n)},z_{2,1},\ldots,z_{u(n),p(n)},w_1,\ldots,w_{p(n)})\Big),$$

where intuitively the variable $z_{i,j}$ is to replace $v_i^{2^j}$ in g_n, and w_i will take the value 2^{2^i}. More formally, h_n is defined as follows:

- replace v_i^k in g_n by $\prod_{j\in J_k} z_{i,j}$, where the set J_k consists of the bits set to 1 in the binary representation of k;
- replace the coefficient 2^{i-1} in the term $\sum_{i=1}^{2^{p(n)}} a(n,\alpha,i)2^{i-1}$ of g_n by $\prod_{j\in J_{i-1}} w_j$, where the set J_{i-1} consists of the bits set to 1 in the binary representation of $i-1$.

The degree of h_n is then polynomially bounded and all the coefficients are among -1, 0 and 1. Note furthermore that the coefficient function is still in PSPACE. Therefore $(h_n) \in \mathrm{VPSPACE}^0_{\mathrm{b}}$, thus $(h_n) \in \mathrm{VP}$ by hypothesis. It remains to replace $z_{i,j}$ by $v_i^{2^j}$ and w_i by 2^{2^i} to show that $(g_n(\bar{v}) = g_n(\bar{x},\bar{y})) \in \mathrm{VP}_{\mathrm{nb}}$, and then to replace $\bar{y}$ by the original constants in order to show that $(f_n) \in \mathrm{VP}_{\mathrm{nb}}$. □

Lemma 11. $\mathrm{VPSPACE}_{\mathrm{b}}$ *contains* VNP.

Proof. Let (HC_n) be the family defined by

$$HC_n(x_{1,1},\ldots,x_{1,n},x_{2,1},\ldots,x_{n,n}) = \sum_{\sigma}\prod_{i=1}^{n} x_{i,\sigma(i)}$$

where the sum is taken over all n-cycles σ over $\{1,\ldots,n\}$. This polynomial counts the number of Hamilton cycles in a graph given by its adjacency matrix. (HC_n) is VNP-complete, see [19] or [14]. Since $\mathrm{VPSPACE}_{\mathrm{b}}$ is closed under p-projections and contains HC_n, the lemma follows. □

C On the Hypothesis that VPSPACE has Small Circuits

In this section, we investigate some consequences of the hypotheses $\mathrm{VPSPACE} = \mathrm{VP}_{\mathrm{nb}}$ and uniform $\mathrm{VPSPACE}^0 =$ uniform $\mathrm{VP}^0_{\mathrm{nb}}$.

Proposition 6. *Under the generalized Riemann hypothesis (GRH),*

$$\mathrm{VP}_{\mathrm{nb}} = \mathrm{VPSPACE} \iff [\mathrm{P/poly} = \mathrm{PSPACE/poly} \textit{ and } \mathrm{VP} = \mathrm{VNP}].$$

Moreover, the implication from right to left holds even without GRH.

Proof. Assume first that P/poly = PSPACE/poly and VP = VNP. By Lemma 10, the equality $\mathrm{VPSPACE} = \mathrm{VP}_{\mathrm{nb}}$ is equivalent to the degree-bounded analogue $\mathrm{VPSPACE}_{\mathrm{b}} = \mathrm{VP}$. Let $(f_n) \in \mathrm{VPSPACE}_{\mathrm{b}}$: its coefficient function is in PSPACE/poly, thus in P/poly by our assumption. Since the set of coefficient functions of VNP families contains $\oplus$P/poly (see [3]), hence P/poly, (f_n) is in fact in VNP. By our assumption again, it is in VP.

For the converse, assume now that $\mathrm{VPSPACE} = \mathrm{VP}_{\mathrm{nb}}$. Again, this is equivalent to $\mathrm{VPSPACE}_{\mathrm{b}} = \mathrm{VP}$. Hence VNP = VP since $\mathrm{VP} \subseteq \mathrm{VNP} \subseteq \mathrm{VPSPACE}_{\mathrm{b}}$ by Lemma 11. It remains to show that a language A in PSPACE/poly belongs in fact to P/poly. A is recognized by a P/poly-uniform family of polynomial-depth boolean circuits, and by Lemma 1 and Proposition 1 there exists a family $(f_n) \in \mathrm{VPSPACE}$ such that on any boolean input $\bar{x} \in \{0,1\}^n$, $f_n(\bar{x}) \in \{0,1\}$ and $f_n(\bar{x}) = 1$ if and only if $\bar{x} \in A$.

By our assumption, $(f_n) \in \mathrm{VP}_{\mathrm{nb}}$, thus there exists a family of polynomial-size arithmetic circuits (C_n), with arbitrary constants, that computes (f_n). In order to evaluate these circuits on boolean inputs with boolean circuits, the problem now is to eliminate the constants. We proceed as in [3]. Let $\bar{y}$ be the constants for the circuit C_n, and call $g_n(\bar{X}, \bar{Y})$ the polynomial computed by C_n where the constants are replaced by the new variables $\bar{Y}$. Thus $g_n(\bar{X}, \bar{y}) = f_n(\bar{X})$, therefore the system S of equations in $\bar{Y}$ defined by

$$S = \big(g_n(\bar{x}, \bar{Y}) = f_n(\bar{x})\big)_{\bar{x} \in \{0,1\}^n}$$

has a solution $\bar{y}$ over $\mathbb{C}$. All the equations in this system have integer coefficients, degree bounded by $2^{q(n)}$ and weight by $2^{2^{q(n)}}$ for some polynomial q, where the weight of a polynomial is the sum of the absolute value of its coefficients.

By Theorem 4.4 of [3, p. 64], assuming GRH there exists a prime number $p \leq 2^{n^2 q(n)}$ such that S has a solution over $\mathbb{F}_p$. There indeed exists such a $p \leq a$ as soon as

$$\frac{\pi(a)}{d^{O(n)}} > \sqrt{a} \log(wa),$$

where d and w are bounds on the degree and weight of the equations respectively, and $\pi(a)$ is the number of primes $\leq a$. Thus there exists a polynomial-size arithmetic circuit over $\mathbb{F}_p$ computing the polynomial $g_n(\bar{X}, \bar{y}')$ and this polynomial takes the same values as $f_n(\bar{X})$ on boolean inputs.

Note that the size of p is polynomial, and a solution $\bar{y}'$ of this system S over $\mathbb{F}_p$ also has polynomial size. Therefore a polynomial-size boolean circuit working modulo p can now easily compute the value of $g_n(\bar{X}, \bar{y}')$ over $\mathbb{F}_p$. This boolean circuit has the same value on boolean inputs as f_n. Hence $A \in$ P/poly, and the announced result is proved. □

We now turn in the next proposition to the most uniform version of the hypothesis, which is stronger than that of Proposition 6. For the proof, we need two definitions from [15] and [14].

Definition 5. *The* formal degree *of an arithmetic circuit C is the formal degree of its output gate, where the formal degree of a gate is defined recursively:*

- *the formal degree of an input gate is 1;*
- *the formal degree of a $+$-gate or a $-$-gate is the maximum of the formal degrees of its inputs;*
- *the formal degree of a $\times$-gate is the sum of the formal degrees of its inputs.*

Definition 6. *The class* VP^0 *is the set of families of polynomials computed by a family of constant-free (i.e. using only 1 as a constant) polynomial-size arithmetic circuits of polynomial formal degree.*

Proposition 7.

$$\text{Uniform } \mathrm{VPSPACE}^0 = \text{uniform } \mathrm{VP}^0_{\mathrm{nb}} \Longrightarrow \mathrm{PSPACE} = \text{P-uniform NC}.$$

Proof. Let us first prove that the hypothesis implies P = PSPACE. Let A be a PSPACE language: it is decided by a uniform family of polynomial-depth boolean circuits. By Lemma 1 and Proposition 1, we obtain a family of polynomials $(f_n) \in$ uniform $\mathrm{VPSPACE}^0$ that agrees with the boolean circuits on boolean inputs, i.e.,

$$\forall \bar{x} \in \{0,1\}^n, f_n(\bar{x}) \in \{0,1\} \text{ and } [f_n(\bar{x}) = 1 \Longleftrightarrow \bar{x} \in A].$$

By our assumption, $(f_n) \in$ uniform $\mathrm{VP}^0_{\mathrm{nb}}$ so that there exists a uniform family of polynomial-size arithmetic circuits that computes (f_n). Of course, on boolean inputs such circuits can be evaluated in polynomial time (working modulo 2 to avoid overflows). This implies that PSPACE = P.

Let us now prove that $\oplus\mathrm{P} \subseteq$ P-uniform NC under the hypothesis that uniform $\mathrm{VPSPACE}^0$ = uniform $\mathrm{VP}^0_{\mathrm{nb}}$. It is enough to show that the $\oplus$P-complete language $\oplus$HamiltonPath (the problem of deciding whether there is an odd number of Hamilton paths in a graph, see [16, p. 448]) is in P-uniform NC. For a graph given by its boolean adjacency matrix $(a_{i,j})$ (where $a_{i,j} = 1$ iff there is an edge between i and j), the number of Hamilton paths is

$$\sum_{1 \leq j < k \leq n} \sum_{\sigma \in S_{j,k}} \prod_{i=1}^{n-1} a_{i,\sigma(i)},$$

where $S_{j,k}$ is the set of all the n-cycles $\sigma \in \mathcal{S}_n$ beginning in j and ending in k (j is different from k in order to count paths in the graph and not cycles, and j is smaller than k in order not to count twice each path, which would trivialize the problem $\oplus$HamiltonPath). The polynomial

$$p_n(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{j<k} \sum_{\sigma \in S_{j,k}} \prod_{i=1}^{n-1} x_{i,\sigma(i)}$$

therefore outputs the number of Hamilton paths on the boolean encoding $x_{1,1} \ldots x_{1,n} x_{2,1} \ldots x_{n,n}$ of a graph G. This family of polynomials (p_n) is easily seen to be in uniform $\mathrm{VPSPACE}^0$, has polynomially bounded degree, and its evaluation modulo 2 provides the answer to the question "$G \in \oplus$HamiltonPath?".

By our assumption, $(p_n) \in$ uniform $\mathrm{VP}^0_{\mathrm{nb}}$ so that there exists a P-uniform family of polynomial-size arithmetic circuits (C_n) that computes (p_n). We are going to build a family of circuits (D_n) that computes a family of polynomials $(q_n) \in \mathrm{VP}^0$ such that on boolean inputs, p_n and q_n have the same parity. Note that despite the polynomial bound on its degree, (p_n) needs not be already in VP^0 because the formal degree of C_n needs not be polynomial (indeed, constants of exponential size might be computed by C_n). This is why we cannot directly evaluate C_n in parallel with the algorithm of [15].

The idea here is that we can compute only the remainder modulo 2 of the constants because we are only interested in the result modulo 2. D_n is then built from C_n as follows. First, note that p_n has degree $n-1$. We compute each homogeneous component separately: each gate α of C_n is split into $n-1$ gates $\alpha_1, \ldots, \alpha_{n-1}$, the gate α_i computing the homogeneous component of degree i of α. The homogeneous components of degree 0 (i.e. the constants) are not computed, only their remainder modulo 2 is taken into account. In other words, we replace an even constant by the constant 0, and an odd one by 1. The P-uniformity remains because we can compute in polynomial time the the constants modulo 2. The last step of D_n is to compute the sum of the homogeneous components of the output gate.

It is easy and well known how to compute these homogeneous components at each step, while keeping a polynomial circuit size: we merely discard the homogeneous components of degree $> n-1$. With this construction, it is clear that p_n and q_n coincide modulo 2, that the construction is P-uniform, and that the formal degree of D_n is at most $n-1$ because there is no constant in the circuit any more. Hence $(q_n) \in \mathrm{VP}^0$.

In order to decide $\oplus$HamiltonPath, we therefore only have to compute the value of q_n modulo 2 on the given input, that is, to evaluate a P-uniform circuit of polynomial size $s(n)$ and polynomially bounded formal degree $n-1$. Theorem 5.3 of [15] tells us that such a circuit can be evaluated modulo 2 by a logspace-uniform algorithm in parallel time $O(\log(s(n)) \log(ns(n)))$, i.e. $O(\log(n)^2)$, and with $O(n^2)$ processors, thus placing $\oplus$P in P-uniform NC^2.

Hence, assuming that uniform $\mathrm{VPSPACE}^0 =$ uniform $\mathrm{VP}^0_{\mathrm{nb}}$ we have proved that

$$\mathrm{PSPACE} = \mathrm{P} \subseteq \oplus\mathrm{P} \subseteq \text{P-uniform } \mathrm{NC}^2.$$

Note that this construction does not seem to be logspace uniform because evaluating the constants modulo 2 is a P-complete problem.

Since we construct a circuit family which is only polynomial-time uniform, one could also use the construction of [20] instead of the parallel algorithm of [15]. Indeed, as pointed out in [15], the construction of [20] can be performed in polynomial time. □

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

D Closure Properties

In the same spirit as Lemma 11 from Appendix B but for the unbounded version, the following lemma is clear by Proposition 1.

Lemma 12. Uniform VPSPACE^0 *is closed under big sums and big products.*

We can even make sums and products over a set more complicated than $\{0,1\}$, as proven in the following lemma.

Lemma 13. *Let A be a language in* PSPACE*, $(f_n(\bar{x},\bar{y}))$ a family in* uniform VPSPACE^0 *and $p(n)$ a polynomial, where $|\bar{y}| = p(n)$. Then the families $(g_n(\bar{x}))$ and $(h_n(\bar{x}))$ defined as follows are in* uniform VPSPACE^0*.*

$$g_n(\bar{x}) = \sum_{\bar{\epsilon}\in A^{=p(n)}} f_n(\bar{x},\bar{\epsilon}) \text{ and } h_n(\bar{x}) = \prod_{\bar{\epsilon}\in A^{=p(n)}} f_n(\bar{x},\bar{\epsilon}).$$

Proof. It is enough to use Lemma 12 since we have

$$\sum_{\bar{\epsilon}\in A^{=p(n)}} f_n(\bar{x},\bar{\epsilon}) = \sum_{\bar{\epsilon}\in\{0,1\}^{p(n)}} \chi_A(\bar{\epsilon}) f_n(\bar{x},\bar{\epsilon}), \text{ and}$$

$$\prod_{\bar{\epsilon}\in A^{=p(n)}} f_n(\bar{x},\bar{\epsilon}) = \prod_{\bar{\epsilon}\in\{0,1\}^{p(n)}} [\chi_A(\bar{\epsilon}) f_n(\bar{x},\bar{\epsilon}) + (1-\chi_A(\bar{\epsilon}))],$$

where χ_A, the characteristic function of A, is in uniform VPSPACE^0 by Lemma 1 and Proposition 1 since A is decided by a uniform family of boolean circuits of polynomial depth. □

E Proof of Proposition 2

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 agebraic 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 and the size of its coefficients is $(nd)^{O(1)}2^i$. Therefore, by Theorem 1 there are at most $(2^d)^{O(n)}$ possible outcomes for the tests of level i, and they are moreover enumerable in space $(nd)^{O(1)}$. Therefore 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 an execution 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 $(2^{dO(n)})^d = 2^{d^2O(n)}$, as claimed in the statement of the theorem.