Valiant's model: from exponential sums to exponential products

Sylvain Perifel, Pascal Koiran

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL nº 5668

Valiant's model: from exponential sums to exponential products

Pascal Koiran
Sylvain Perifel

June 2006

Rapport de recherche Nº 2006–21

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

Valiant's model: from exponential sums to exponential products

Pascal Koiran
Sylvain Perifel

June 2006

Abstract

We study the power of big products for computing multivariate polynomials in a Valiant-like framework. More precisely, we define a new class $\mathrm{V\Pi P}^0$ as the set of families of polynomials that are exponential products of easily computable polynomials. We investigate the consequences of the hypothesis that these big products are themselves easily computable. For instance, this hypothesis would imply that the nonuniform versions of P and NP coincide. Our main result relates this hypothesis to Blum, Shub and Smale's algebraic version of P versus NP. Let K be a field of characteristic 0. Roughly speaking, we show that in order to separate P_K from NP_K using a problem from a fairly large class of "simple" problems, one should first be able to show that exponential products are not easily computable. The class of "simple" problems under consideration is the class of NP problems in the structure $(K, +, -, =)$, in which multiplication is not allowed.

Keywords: Algebraic complexity, Valiant's model, Blum–Shub–Smale's model, big products.

Résumé

Cet article étudie la puissance des gros produits pour le calcul de polynômes à plusieurs variables dans le cadre de la théorie de Valiant. Plus précisément, nous définissons pour cela une nouvelle classe $\mathrm{V\Pi P}^0$ de familles de polynômes : il s'agit des produits de taille exponentielle de polynômes facilement calculables. Nous étudions les conséquences de l'hypothèse que ces gros produits sont eux-mêmes facilement calculables. Par exemple, cela impliquerait que les versions non-uniformes de P et NP coïncident. Le résultat principal est un lien avec les classes algébriques P et NP du modèle BSS sur un corps K de caractéristique nulle. On pourrait l'énoncer ainsi : si nous voulons séparer P_K de NP_K grâce à des problèmes issus d'un ensemble important de problèmes « simples », il faut d'abord être capable de montrer que nos gros produits ne sont pas facilement calculables. L'ensemble des problèmes « simples » en question est NP sur la structure $(K, +, -, =)$, dans laquelle la multiplication n'est pas autorisée.

Mots-clés: Complexité algébrique, modèle de Valiant, modèle BSS, gros produits.

1 Introduction

Valiant's model. In the framework of Valiant's theory, which goes back to [18], the objects of interest are families of multivariate polynomials. The complexity of such families can be measured by the size of arithmetic circuits which compute them. Two main complexity classes were introduced : VP, whose elements are families of polynomials computed by arithmetic circuits of polynomial size and polynomially bounded degree, and VNP. A VNP family is obtained from a VP family by a summation of (possibly) exponential size, and a central open question is whether VP and VNP coincide. For a long time, these two classes were almost the only classes studied in Valiant's theory. One exception is the class VQP of polynomials computed by arithmetic circuits of quasi-polynomial size of polynomially bounded degree. More recently, new classes were defined and studied by Malod [13]. Of particular interest for us is his class $\mathrm{VP}^0_{\mathrm{nb}}$. In contrast to VP, arbitrary constants are not allowed, and the degrees of polynomials are not bounded.

In this paper we define a new class, called $\mathrm{V\Pi P}^0$, which is obtained from $\mathrm{VP}^0_{\mathrm{nb}}$ by computing products of (possibly) exponential size. By definition $\mathrm{VP}^0_{\mathrm{nb}}$ is included in $\mathrm{V\Pi P}^0$, and we conjecture that this inclusion is strict. Some support for this conjecture is provided by the following observation : if $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$ the polynomial family

$$P_d = \prod_{i=0}^{d-1} (X - i) \tag{1}$$

is easy to compute, i.e., can be computed by a family of arithmetic circuits of size polynomial in $\log d$. However, there is in algebraic complexity theory a fairly old conjecture that this family is hard to compute [7, 12]. Even more compelling support for our conjecture is provided by Theorem 1, which shows that the non-uniform versions of P and NP coincide if $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$, that is, if big products are computable by polynomial size circuits.

The goal of this paper is not merely to define yet another complexity class. Indeed, as explained below the study of $\mathrm{V\Pi P}^0$ leads to meaningful results about the complexity of *decision* problems. This paper is therefore in the same spirit as [9], where it is shown that certain sequences of integers become easy to compute if certain classes of polynomial families coincide.

Blum-Shub-Smale model. One crucial difference between this second model of algebraic computation and Valiant's model is the focus on decision (rather than evaluation) problems. Precise definitions will be given in section 2. In this introduction we will just recall that there is for each field a version of the classical P versus NP problem. In particular, for the field of complex numbers there is a very natural "$\mathrm{P}_{\mathbb{C}} = \mathrm{NP}_{\mathbb{C}}$?" problem, which has remained open since [3]. In order to separate $\mathrm{P}_{\mathbb{C}}$ from $\mathrm{NP}_{\mathbb{C}}$, it is of course sufficient to exhibit a "well chosen" problem A which belongs to $\mathrm{NP}_{\mathbb{C}}$ but not to $\mathrm{P}_{\mathbb{C}}$. One natural choice would be to try $A = \mathrm{FEAS}_{\mathbb{C}}$, where $\mathrm{FEAS}_{\mathbb{C}}$, the feasibility problem for systems of polynomial equations, is the canonical $\mathrm{NP}_{\mathbb{C}}$-complete problem. One insight from Shub and Smale [17] was that there are much more elementary-looking $\mathrm{NP}_{\mathbb{C}}$ problems that do not seem to belong to $\mathrm{P}_{\mathbb{C}}$. Shub and Smale's candidate is the problem Twenty Questions, which can be defined as follows : given a complex number x and an integer d written in binary, decide whether x is an integer in the set $\{0, 1, \ldots, d-1\}$. It is not difficult to see that this problem is in $\mathrm{NP}_{\mathbb{C}}$ (hint : guess the binary decomposition of x). Shub and Smale gave compelling evidence that this problem does not belong to $\mathrm{P}_{\mathbb{C}}$, but no conclusive proof could be obtained. In hindsight, this lack of definitive results is not surprising. Indeed, to decide whether an input to Twenty Questions should be accepted it suffices to evaluate the polynomial P_d at $X = x$, and to compare the result to 0. In order to show that Twenty Questions is not in $\mathrm{P}_{\mathbb{C}}$, one must therefore show that the family P_d is hard to compute. As explained above, this is a fairly longstanding open problem[1] which actually predates [17].

In this paper we investigate the following question : are there other examples of "simple" problems which might be used to separate $\mathrm{NP}_{\mathbb{C}}$ from $\mathrm{P}_{\mathbb{C}}$? The class of "simple" problems that we have in mind is $\mathrm{NP}_{(\mathbb{C},+,-,=)}$. This is the class of NP problems over the set of complex numbers endowed with addition, subtraction, and equality tests (there is therefore no multiplication in this structure). It contains Twenty Questions and many other natural problems (for instance, Subset Sum). Our main result, Theorem 2, is established in section 5 : we show that if $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$ then $\mathrm{NP}_{(\mathbb{C},+,-,=)}$ is contained in $\mathbb{P}_{(\mathbb{C},+,-,\times,=)}$, the non-uniform version of $\mathrm{P}_{\mathbb{C}}$. Here, the non-uniformity is only due to the fact that (in keeping with the

[1] The computation model of [7] and [12] is non-uniform, but Shub and Smale's is uniform. It doesn't seem, however, that adding a uniformity requirement would be of much help in showing that the family P_d is hard to compute.

tradition set by Valiant) the classes $\mathrm{VP}^0_{\mathrm{nb}}$ and $\mathrm{V\Pi P}^0$ are non-uniform. One could equally well work with uniform versions of $\mathrm{VP}^0_{\mathrm{nb}}$and $\mathrm{V\Pi P}^0$, and arrive instead at the inclusion $\mathrm{NP}_{(\mathbb{C},+,-,=)} \subseteq \mathrm{P}_{\mathbb{C}}$.

We hope that this paper will help put the focus back from decision problems to evaluation problems. Indeed, we have shown that in order to prove good lower bounds for problems in a fairly large class of decision problems, one must first be able to prove good lower bounds for a related class of evaluation problems. It is a natural question whether the study of evaluation problems can shed light not only on the problem "$\mathrm{NP}_{(\mathbb{C},+,-,=)} \subseteq \mathrm{P}_{\mathbb{C}}$?", but also on the full "$\mathrm{P}_{\mathbb{C}} = \mathrm{NP}_{\mathbb{C}}$?" problem, or on the "$\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$?" problem. This question will be investigated in a forthcoming paper.

The present paper is the full version of [11].

2 Notations

Our polynomials will be multivariate, and for notational simplicity a tuple of indeterminates will be denoted $\bar{x}$ instead of $(x_1, \ldots, x_{u(n)})$. We will use the Greek letter $\bar{\epsilon}$ to emphasize that we are using a tuple of boolean variables, i.e. $\bar{\epsilon} \in \{0,1\}^{u(n)}$. However, depending on the context $\bar{x}$ will also denote a boolean word when we are dealing with boolean problems.

2.1 Boolean complexity classes

We will not offend the reader by defining the boolean classes P and NP. Let us only recall the definitions of their nonuniform versions P/poly and NP/poly. P/poly is defined equivalently in terms of circuits or machines : this is the set of boolean langugages recognized by a family of boolean circuits of polynomial size. Alternatively, this is also the set of languages recognized by a Turing machine working in polynomial time with the help of a polynomial size advice function (hence the name P/poly, see [8]).

NP/poly, the nonuniform version of NP, is the set of languages recognized by polynomial time nondeterministic Turing machine with the help of a polynomial size advice function. Equivalently, it is easily seen to be the nondeterministic counterpart of P/poly, that is to say : $L \in$ NP/poly if and only if there exist $A \in$ P/poly and a polynomial $p(n)$ such that

$$\bar{x} \in L \iff \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(\bar{x}, \bar{y}) \in A.$$

If A is a language and k a nonnegative integer, $A^{=k}$ denotes the set of words of A of size k.

Another class used in this paper is coRP. It is the set of languages recognized in polynomial time by randomized Turing machines with one-sided error. For more details on boolean complexity, we refer the reader to [14] for instance.

2.2 Algebraic circuits

In this section we recall the definitions of the non-uniform classes $\mathbb{P}_{(K,+,-,\times,=)}$ and $\mathbb{NP}_{(K,+,-,\times,=)}$, where K is an arbitrary field. These two classes are the non-uniform versions of the classes P_K and NP_K defined by Blum, Shub and Smale [3, 2]. Following [15], we will use families of algebraic circuits to recognize languages over K, that is, subsets of $K^\infty = \bigcup_{n\geq 0} K^n$.

An algebraic circuit (understood over $(K, +, -, \times, =)$) 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 K^n to K. On input $\bar{u} \in K^n$ 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. We assume without loss of generality that the output is a test gate. The value returned by the circuit is therefore 0 or 1.

Finally, the class $\mathbb{P}_{(K,+,-,\times,=)}$ is the set of languages $L \subseteq K^\infty$ such that there exists a tuple $\bar{a} \in K^p$ and a polynomial-size circuit family (C_n) satisfying the following condition : C_n has exactly $n+p$ inputs, and for any $\bar{x} \in K^n$, $\bar{x} \in L \Leftrightarrow C_n(\bar{x}, \bar{a}) = 1$. Note that $\bar{a}$ plays the role of the machine constants of [2, 3]. The uniform class P_K of [2, 3] can be obtained from $\mathbb{P}_{(K,+,-,\times,=)}$ by adding a uniformity requirement on the family (C_n). In this paper we will stick to non-uniform classes.

Furthermore, $\mathbb{NP}_{(K,+,-,\times,=)}$ is the class of languages L such that there exists a language $A \in \mathbb{P}_{(K,+,-,\times,=)}$ and a polynomial $p(n)$ satisfying

$$\bar{x} \in L \iff \exists \bar{y} \in K^{p(|\bar{x}|)}.(\bar{x}, \bar{y}) \in A.$$

We also define a version $\mathbb{DNP}_{(K,+,-,\times,=)}$ ('D' stands for "digital"), where nondeterminism is allowed only on boolean tuples :

$$\bar{x} \in L \iff \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(\bar{x}, \bar{y}) \in A.$$

We will also need to compute over the structure $(K, +, -, =)$, where multiplication is not allowed. An algebraic circuit over $(K, +, -, =)$ is defined as above, except that there are no $\times$-gate and that there is a new type of gates, called selection gates. A selection gates is of indegree 3. Its value on input (x, y, z) is x if $z = 0$, and y otherwise. The role of these gates is to simulate "if then else" statements. These gates are not needed for the structure $(K, +, -, \times, =)$ since "if then else" statements can be simulated using multiplication (for instance, by the subcircuit $[z = 0?] \times x + (1 - [z = 0?]) \times y$). The classes $\mathbb{P}_{(K,+,-,=)}$ and $\mathbb{NP}_{(K,+,-,=)}$ are defined in the same way as $\mathbb{P}_{(K,+,-,\times,=)}$ and $\mathbb{NP}_{(K,+,-,\times,=)}$. We could define $\mathbb{DNP}_{(K,+,-,=)}$ as well, but the first author has shown in [10] that $\mathbb{DNP}_{(K,+,-,=)} = \mathbb{NP}_{(K,+,-,=)}$, i.e., only digital nondeterminism is enough over the structure $(K, +, -, =)$.

2.3 Arithmetic circuits

In Valiant's model, we compute polynomials instead of recognizing languages. A book-length treatment of this topic can be found in [4]. In our framework, which, as explained in the introduction, is not the original one, we require the underlying structure to be a field of characteristic 0, and do not allow arbitrary constants (apart from the constant 1) in our circuits. Hence we compute polynomials $f_n \in \mathbb{Z}[x_1, \ldots, x_{u(n)}]$. Furthermore, we have no restriction on the degree of the polynomials. This formalism was introduced and studied in [13].

An arithmetic circuit is the same as an algebraic circuit over $(K, +, -, \times, =)$, but test gates are not allowed. That is to say we have indeterminates $x_1, \ldots, x_{u(n)}$ as input, $+$, $-$ and $\times$-gates, and we therefore compute polynomials with integer coefficients.

The polynomial computed by an arithmetic circuit is defined in the usual way. Thus a family (C_n) of arithmetic circuits computes a family (f_n) of polynomials, $f_n \in \mathbb{Z}[x_1, \ldots, x_{u(n)}]$. The class $\mathrm{VP}^0_{\mathrm{nb}}$ 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.

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 Big products

We introduce here the new class $\mathrm{V\Pi P}^0$, where exponential products are allowed. This is very much inspired by the class VNP, but sums are replaced by products (and, as explained before, constants different from 1 are not allowed, and there is no restriction on the degree).

Definition 1 *The class $\mathrm{V\Pi P}^0$ is the set of families of polynomials $(g_n(\bar{x}))$ such that there exists a family $(f_n(\bar{x}, \bar{y})) \in \mathrm{VP}^0_{\mathrm{nb}}$ satisfying the relation :*

$$g_n(\bar{x}) = \prod_{\bar{\epsilon} \in \{0,1\}^{|\bar{y}|}} f_n(\bar{x}, \bar{\epsilon}).$$

Example 1 *The family $(g_n(X))$ defined by $g_n(X) = \prod_{i=0}^{2^n-1}(X-i)$ is in $\mathrm{V\Pi P}^0$. To see this, let $(f_n(X,\bar{\epsilon}))$ be the family*

$$f_n(X,\bar{\epsilon}) = X - \sum_{j=1}^{n} \epsilon_j 2^{j-1}.$$

Then $(f_n) \in \mathrm{VP}^0_{\mathrm{nb}}$ and $g_n(X) = \prod_{\bar{\epsilon}\in\{0,1\}^n} f_n(X,\bar{\epsilon})$.

Note that $g_n = P_{2^n}$, where P_{2^n} is defined by (1). This polynomial can therefore be computed by a circuit of size polynomial in n if $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$. In fact a more general property holds true : if $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$ the family (P_d) is easy to compute. Indeed, once we know how to evaluate efficiently P_d when d is a power of 2, we can also evaluate efficiently for an arbitrary d thanks to the relation $P_{d+2^n}(X) = P_d(X)P_{2^n}(X-d)$. This observation gives some plausibility to the conjecture $\mathrm{VP}^0_{\mathrm{nb}} \neq \mathrm{V\Pi P}^0$. Additional support is provided by Theorem 1 from section 4.

Remark 1 *The underlying field is implicit in the notations $\mathrm{VP}^0_{\mathrm{nb}}$ and $\mathrm{V\Pi P}^0$, and should usually be clear from the context. Note that for the question $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$, there is no ambiguity at all. Indeed, the equality $\mathrm{VP}^0_{\mathrm{nb}} = \mathrm{V\Pi P}^0$ holds true in a field of characteristic 0 if and only if it holds true in all fields of characteristic 0.*

Remark 2 *In the spirit of the polynomial hierarchy in boolean complexity theory, one could define a whole hierarchy of new complexity classes by alternating sums and products. The classes $\mathrm{VP}^0_{\mathrm{nb}}$, $\mathrm{VNP}^0_{\mathrm{nb}}$ (also studied by Malod [13]) and $\mathrm{V\Pi P}^0$ would be the first three classes of this hierarchy.*

Next we present a criterion which enables to make products over a set more complicated than $\{0,1\}^n$.

Lemma 2 *Let $(f_n(\bar{x},\bar{y}))$ be a $\mathrm{VP}^0_{\mathrm{nb}}$ family, and $s(n)$ a function which bounds from above the length of $\bar{y}$, and is itself polynomially bounded (i.e., $s(n) \leq p(n)$ for some polynomial p). Let A be a language in $\mathrm{P/poly}$. There exists a family $(g_n(\bar{l},\bar{x}))$ in $\mathrm{V\Pi P}^0$, where $|\bar{l}| = s(n) - |\bar{y}|$, such that for any tuple $\bar{x}$ of elements of K and any boolean tuple $\bar{l}$ we have :*

$$g_n(\bar{l},\bar{x}) = \prod_{\bar{\epsilon};\ (\bar{l},\bar{\epsilon})\in A^{=s(n)}} f_n(\bar{x},\bar{\epsilon}).$$

Proof. Since $A \in \mathrm{P/poly}$, there exists a family of polynomial size boolean circuits (C_n) deciding A. By Lemma 1, we can simulate this family of boolean circuits by a family of arithmetic circuits. We obtain a family of polynomials $(c_n(\bar{y},\bar{z}))$ in $\mathrm{VP}^0_{\mathrm{nb}}$ such that for any boolean input $(\bar{l},\bar{\epsilon})$ of size n :

$$c_n(\bar{l},\bar{\epsilon}) = \begin{cases} 1 & \text{if } (\bar{l},\bar{\epsilon}) \in A \\ 0 & \text{otherwise.} \end{cases}$$

The family $(h_n(\bar{x},\bar{y},\bar{z}))$ defined by

$$h_n(\bar{x},\bar{y},\bar{z}) = c_{s(n)}(\bar{y},\bar{z})f_n(\bar{x},\bar{z}) + 1 - c_{s(n)}(\bar{y},\bar{z})$$

is therefore in $\mathrm{VP}^0_{\mathrm{nb}}$ and satisfies

$$\prod_{\bar{\epsilon}\in\{0,1\}^{s(n)}} h_n(\bar{x},\bar{l},\bar{\epsilon}) = \prod_{\bar{\epsilon};\ (\bar{l},\bar{\epsilon})\in A^{=s(n)}} f_n(\bar{x},\bar{\epsilon}).$$

□

Note that this lemma is already meaningful when $s(n) = |\bar{y}|$, i.e., when $|\bar{l}| = 0$. The more general statement given here will be useful for the proof of our main theorem.

4 Boolean complexity

In this section we explore the consequences for boolean complexity theory of the assumption that big products are computable by polynomial size circuits. Namely, we prove the following result.

Theorem 1 *If* $\mathrm{V\Pi P}^0 = \mathrm{VP}^0_{\mathrm{nb}}$ *then* $\mathrm{P/poly} = \mathrm{NP/poly}$.

Proof. Let $A \in \mathrm{NP/poly}$. Then there exist a language $B \in \mathrm{P/poly}$ and a polynomial $p(n)$ such that

$$\bar{x} \in A \iff \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(\bar{x},\bar{y}) \in B.$$

Since $B \in \mathrm{P/poly}$, it is decided by a family of polynomial size boolean circuits. These circuits can be simulated by arithmetic ones as in Lemma 1. We obtain a family of polynomials $(f_n(\bar{x},\bar{y}))$, whose value on a boolean input $(\bar{x},\bar{y})$ is 0 if $(\bar{x},\bar{y}) \in B$ and 1 otherwise. This family is in $\mathrm{VP}^0_{\mathrm{nb}}$ because the family of arithmetic circuits has polynomial size.

Now, the products

$$g_n(\bar{x}) = \prod_{\bar{y} \in \{0,1\}^{p(|\bar{x}|)}} f_n(\bar{x},\bar{y})$$

form a $\mathrm{V\Pi P}^0$ family. On any boolean input $\bar{x}$ we have $g_n(\bar{x}) \in \{0,1\}$, and $g_n(\bar{x}) = 0$ iff $\exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(f_n(\bar{x},\bar{y}) = 0)$. In other words,

$$g_n(\bar{x}) = 0 \Leftrightarrow \bar{x} \in A. \tag{2}$$

Under the hypothesis $\mathrm{V\Pi P}^0 = \mathrm{VP}^0_{\mathrm{nb}}$, the family (g_n) is in $\mathrm{VP}^0_{\mathrm{nb}}$. It is therefore computed by polynomial size arithmetic circuits. Deciding whether $\bar{x} \in A$ in nonuniform polynomial time thus amounts to testing in nonuniform polynomial time whether the value of a circuit is zero. It is well known that this can be done in randomized polynomial time coRP by computing modulo random primes (see for instance [16]). The inclusion $\mathrm{coRP} \subset \mathrm{P/poly}$ [1] concludes the proof. □

It follows from (2) that we can decide any problem in NP by testing an appropriate $\mathrm{V\Pi P}^0$ family for zero. This fact will be used in section 5.4.

5 A transfer theorem

We now turn our attention to links with the Blum-Shub-Smale model. The main result of this section, and of the present paper, is the following theorem.

Theorem 2 *If* $\mathrm{V\Pi P}^0 = \mathrm{VP}^0_{\mathrm{nb}}$ *then* $\mathbb{NP}_{(K,+,-,=)} \subseteq \mathbb{P}_{(K,+,-,\times,=)}$.

As in Theorem 1, this connection between $\mathrm{V\Pi P}^0$ and nondeterminism will be obtained by replacing quantifiers by products. However, in $\mathrm{V\Pi P}^0$ the products concern only arithmetic circuits, whereas in $\mathbb{NP}_{(K,+,-,=)}$ the quantifiers concern algebraic circuits (where test gates occur). Therefore, we would like to simulate the computation of an algebraic circuit by an arithmetic one, i.e., to eliminate the test gates. For this purpose, we use boolean circuits as an intermediate step. The latter can indeed be easily simulated by arithmetic circuits by Lemma 1. Doing so requires to deal only with boolean inputs. One part of this problem has already been solved in [10] : boolean nondeterminism already captures $\mathbb{NP}_{(K,+,-,=)}$. It remains to replace the algebraic input $\bar{x} \in K^n$ by a boolean one. This is achieved in the sequel by using mostly techniques which deal with arrangements of hyperplanes. The idea is to replace the algebraic input $\bar{x} \in K^n$ by a point $\bar{q} \in K^n$ of "small" rational coefficients, "close enough" to $\bar{x}$ so that their behaviours will be the same. Now, this rational point can be encoded by boolean tuples, and the whole computation simulated by boolean circuits. "Close enough" means in fact that $\bar{x}$ and $\bar{q}$ belong to the same cell of a suitable arrangement of hyperplanes, i.e., lie on exactly the same hyperplanes of the arrangement. Similar ideas were used in the proofs of the transfer theorems of [5] and [6], which dealt with the structure $(\mathbb{R},+,-,<)$. Note however that the cells of an arrangement as defined below are not the same as in these two papers. Indeed, since we work in an unordered structure, it doesn't make sense to ask whether a point is "above" or "below" a given hyperplane. The only thing that matters is whether the point lies or not on the hyperplane.

Point location in arrangements of hyperplanes is the main ingredient for finding the rational point $\bar{q}$ on input $\bar{x}$. For a given family of hyperplanes, the goal is to build a circuit which outputs the cell of $\bar{x}$. These notions are explained in section 5.1. In section 5.2 we explain how to find the cell of $\bar{x}$ using $\mathrm{V\Pi P}^0$ tests. Then section 5.3 deals with the existence of small rational points in the cell of $\bar{x}$. Finally, these tools are put together in section 5.4 to recognize $\mathbb{NP}_{(K,+,-,=)}$ problems with the help of $\mathrm{V\Pi P}^0$ tests.

5.1 Arrangement of hyperplanes

By *hyperplane* (or *affine* hyperplane) of K^n, we mean a surface (of dimension $n-1$) defined by an affine equation $\sum_i \lambda_i x_i = \mu$. We say that k linear hyperplanes of K^n are *independent* if their intersection has dimension exactly $n-k$. In other words, the k hyperplanes are in general position.

An *arrangement of hyperplanes* is merely a finite family of affine hyperplanes $\mathscr{A} = \{H_i;\ i \in I\}$. This enables us to define an equivalence relation :

$$\bar{x} \sim \bar{y} \text{ iff } \forall i.(\bar{x} \in H_i \iff \bar{y} \in H_i).$$

The equivalence classes are called *cells* of the arrangement. In other words, two points are in the same cell if they belong to exactly the same hyperplanes. A cell is therefore of the form

$$(\bigcap_{i \in J} H_i) \setminus (\bigcup_{j \in J'} H_j)$$

for some subsets J and J' of I. One can assume without loss of generality that the hyperplanes $(H_i)_{i\in J}$ are independent. Notice that the cell of $\bar{x} \in K^n$ is characterized by a maximal set (with respect to inclusion) of independent hyperplanes that contain $\bar{x}$. We will use this characterization later for describing the cells of our arrangement. As outlined at the beginning of section 5, on input $\bar{x} \in K^n$ we want to determine its cell, i.e., to return the indices of these independent hyperplanes.

Let $p(n)$ be a fixed polynomial, and $\mathscr{A}_{p,n}$ the set of all hyperplanes in K^n with integer coefficients of absolute value at most $2^{p(n)}$. We call $\mathscr{H}_p$ the family of all the arrangements $\mathscr{A}_{p,n}$ (where n ranges over $\mathbb{N} \setminus \{0\}$). Section 5.2 explains how to build a family of polynomial-size circuits with $\mathrm{V\Pi P}^0$ tests which, on input $\bar{x} \in K^n$, output the cell of $\bar{x}$ in the arrangement $\mathscr{A}_{p,n}$ (this is called "point location" in the arrangement).

5.2 Point location

The goal of this section is to build an algebraic circuit with $\mathrm{V\Pi P}^0$ tests, which on input $\bar{x} \in K^n$ returns its cell. We first define formally circuits with $\mathrm{V\Pi P}^0$ tests. Then we prove that the point location problem can be solved efficiently using $\mathrm{V\Pi P}^0$ tests.

Definition 2 *A family of algebraic circuits with* $\mathrm{V\Pi P}^0$ *tests is a family* $(f_n(\bar{x})) \in \mathrm{V\Pi P}^0$ *together with a family* (C_n) *of algebraic circuits, where* C_n *is endowed with gates labeled by "*$f_n(\bar{y}) = 0$ *?" (the subscript* n *has to be the same for* f_n *and* C_n*). These gates are of indegree* $|\bar{y}|$ *and output 0 if the test fails (i.e.* f_n *evaluated on the inputs of the gate is nonzero), 1 otherwise.*

The class $\mathbb{P}_{(K,+,-,\times,=)}(\mathrm{V\Pi P}^0)$ *is the set of languages recognized by a family of polynomial size algebraic circuits with* $\mathrm{V\Pi P}^0$ *tests.*

By adding some "selection variables", it is not hard to see that in fact any constant number of $\mathrm{V\Pi P}^0$ families can be tested (instead of only one) and still we stay in $\mathbb{P}_{(K,+,-,\times,=)}(\mathrm{V\Pi P}^0)$. For instance, two $\mathrm{V\Pi P}^0$ families will be used in section 5.4 : one family to perform a point location task, and the other family to decide a (classical) NP problem. We now explain how to solve the point location problem using $\mathrm{V\Pi P}^0$ tests.

Proposition 1 *Let* $(\mathscr{H}_p)$ *be the family of arrangements of hyperplanes whose coefficients are integers bounded by* $2^{p(n)}$ *in absolute value (this family was defined at the end of section 5.1). There exists a family* (C_n) *of polynomial size algebraic circuits with* $\mathrm{V\Pi P}^0$ *tests that, on input* $\bar{x} \in K^n$*, output the indices of* m *independent hyperplanes that characterize the cell of* $\bar{x}$*.*

Proof. The idea of the algorithm is simple : we maintain a "search space" E which locates $\bar{x}$ as accurately as possible. At the beginning we have no information, and we let $E = K^n$. At each subsequent step, we find (if it exists) the first hyperplane H of our arrangement that refines E, i.e., $\bar{x} \in H$ and $\dim(E \cap H) < \dim E$. At most n steps are therefore enough, and as the description of the cell of $\bar{x}$ we return the indices of the successive hyperplanes met during this process. We will explain below how to find the first hyperplane refining E with the help of $\mathrm{V\Pi P}^0$ tests. Let us first sum up the algorithm :

- $E \leftarrow K^n$;
- $L \leftarrow \emptyset$;
- $R \leftarrow \{H \in \mathscr{A} : \bar{x} \in H\}$;
- while $R \neq \emptyset$ do
  . let H_0 be the first hyperplane of R
  . $L \leftarrow L \cup \{H_0\}$
  . $E \leftarrow E \cap H_0$
  . $R \leftarrow \{H \in \mathscr{A} : \bar{x} \in H \text{ and } E \cap H \neq E\}$
- return L.

Note that $E = \bigcap_{H \in L} H$, thus keeping track of L (a list of hyperplanes, or actually of their indices) is enough to determine E.

Finding the first hyperplane refining E is done by binary search, thanks to $\mathrm{V\Pi P}^0$ tests. The list L describing E contains at most n indices, all of size polynomial in n. We store this list in a polynomial number of variables $l_1, \ldots, l_{q(n)}$, representing the boolean encoding of these indices.

At each step, let A be the set of indices of hyperplanes that do not contain E. If f_i is the equation of H_i, the polynomial

$$g(\bar{l}, \bar{x}) = \prod_{i<j \text{ and } i \in A} f_i(\bar{x})$$

vanishes if and only if the first hyperplane refining E has its index smaller than j. By making j vary, we can thus find this hyperplane via binary search in a number of steps which is logarithmic in the number of hyperplanes, i.e., polynomial in n.

We now explain why this product is in $\mathrm{V\Pi P}^0$. With boolean inputs $l_1, \ldots, l_{q(n)}$ and i, we can compute the equation of H_i and test by a simple rank calculation whether H_i has nontrivial trace over E. This is done by a boolean circuit of polynomial size, for instance by Gaussian elimination. Now, Lemma 2 ensures that this product is in $\mathrm{V\Pi P}^0$.

In a polynomial number of $\mathrm{V\Pi P}^0$ tests, we therefore find the first hyperplane refining E. We then proceed with the next step : after at most n steps we have completely characterized the cell of $\bar{x}$. We output the list L of the successive hyperplanes found. This concludes the proof of Proposition 1. $\square$

5.3 Small rational points

Given a description of the cell of an input $\bar{x}$ we aim at finding a small rational point in it, so as to work on boolean rather than algebraic inputs. We begin by a simple lemma on the size of rational points. Then we show in Lemma 5 how to find a rational point of small size in a given cell.

We say that a rational number q has *size at most k* if its numerator as well as its denominator are both of absolute value at most 2^k. The following lemma is straightforward.

Lemma 3 *Let α and β be two rational numbers of size $\leq t$ and $\leq t'$ respectively. Then*
- *$\alpha\beta$ is of size $\leq t + t'$;*
- *$\alpha + \beta$ is of size $\leq t + t' + 1$.*

In particular, if M is a matrix of size $n \times m$ whose coefficients are rationals of size $\leq t$, and x a vector of size n whose coefficients are rationals of size $\leq t'$, then Ax is a vector of $\mathbb{Q}^m$ whose coefficients are rationals of size $\leq n(t + t') + n - 1$.

A point $\bar{q}$ is in the same cell as $\bar{x}$ if it is in the same intersection of hyperplanes, and also outside the same hyperplanes as $\bar{x}$. That is why we need the following lemma, which exhibits a point outside a given set of hyperplanes.

Lemma 4 *Let $\mathscr{A}$ be a family of hyperplanes whose coefficients are integers bounded in absolute value by k. Then the point $\bar{q}$ of coordinates $q_i = (k+1)^i$ (for $i = 1, \ldots, n$) does not belong to any of the hyperplanes of $\mathscr{A}$.*

Proof. Let $f(\bar{x}) = \sum_{i=1}^{n} \alpha_i x_i + b$ be the equation of a hyperplane H of $\mathscr{A}$. For $a \in \mathbb{Z}$, let $a^+ = \max(0, a)$ and $a^- = \max(0, -a)$. Note that $a = a^+ - a^-$, and that $0 \leq a^-, a^+ \leq k$. We define $f^+(\bar{x}) = \sum_{i=1}^{n} \alpha_i^+ x_i + b^+$ and $f^-(\bar{x}) = \sum_{i=1}^{n} \alpha_i^- x_i + b^-$.

Then $\bar{q}$ is in H if and only if $f^-(\bar{q}) = f^+(\bar{q})$, i.e., $\sum_i \alpha_i^-(k+1)^i + b^+ = \sum_i \alpha_i^+(k+1)^i + b^-$. By unicity of the decomposition in base $(k+1)$ this is equivalent to the conditions : $b^+ = b^-$ and $\forall i, \alpha_i^- = \alpha_i^+$. Hence $b = 0$ and $\alpha_i = 0$ for all i. This is in contradiction with the hypothesis that H is a hyperplane. □

The next lemma shows that a point with small rational coordinates exists in a given cell, and can easily be found (by a boolean circuit of polynomial size). Recall that, as explained at the end of section 5.1, $\mathscr{H}_p$ is the family of arrangements whose hyperplanes have integer coefficients bounded by $2^{p(n)}$ in absolute value. We only sketch the proof since the details are only routine calculations.

Lemma 5 *For the family of arrangements $\mathscr{H}_p$, there exists a family (C_n) of boolean circuits of size polynomial in n satisfying the following property : C_n takes as input the indices of $m \leq n$ independent hyperplanes of K^n, and outputs a vector $\bar{q}$ such that :*
- *$\bar{q}$ is in the cell defined by the m hyperplanes ;*
- *$\bar{q}$ has rational coordinates, all of them of size polynomial in n.*

Proof. Let E be the intersection of the m hyperplanes : this is an affine subspace of K^n of dimension $n-m$. The cell is of the form $E \setminus U$, where U is a finite (and possibly empty) union of affine subspaces of dimension $n-m-1$. The equation of E is of the form $Ax = b$ for some $m \times n$ matrix A. We can find in polynomial time a set of m columns of A of rank m. Assume for notational simplicity that these columns are the m first ones. Let $\phi : K^{n-m} \to E$ be the affine map which sends $(x_{m+1}, \ldots, x_n)$ to $(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$, where $(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$ is the only point of E whose $n-m$ last coodinates are $(x_{m+1}, \ldots, x_n)$. The linear part of ϕ is an isomorphism of linear spaces. The coefficients of ϕ are obtained from those of A and b by solving a linear system of equations. They are therefore rational numbers of size polynomial in n. If H is a hyperplane of our arrangement with a nontrivial intersection with E, $\phi^{-1}(E \cap H)$ is a hyperplane of K^{n-m} whose coefficients are integers of size polynomial in n.

Furthermore, thanks to Lemma 4 we can construct a point $\bar{q} \in K^{n-m}$ whose coefficients are integers of size polynomial in n, and which lies on none of the $\phi^{-1}(E \cap H)$. Now, by Lemma 3 $\phi(\bar{q})$ has rational coefficients of size polynomial in n, and it is in the cell. □

5.4 Deciding $\mathbb{NP}_{(K,+,-,=)}$problems

We are now ready for the main theorem of this section : $\mathbb{NP}_{(K,+,-,=)}$ problems are decided by polynomial size algebraic circuits with $\mathrm{V\Pi P}^0$ tests.

Theorem 3 *Let K be a field of characteristic zero. Then*

$$\mathbb{NP}_{(K,+,-,=)} \subseteq \mathbb{P}_{(K,+,-,\times,=)}(\mathrm{V\Pi P}^0).$$

If big products are computable by arithmetic circuits of polynomial size, one can efficiently simulate $\mathrm{V\Pi P}^0$ tests with algebraic circuits. Theorem 2 therefore follows immediately from Theorem 3.
Proof. (of Theorem 3)
The outline of the proof is as follows. First we determine the cell of $\bar{x}$. Then we construct in polynomial time a small rational point $\bar{q}$ in the cell. Deciding whether $\bar{q}$ is a positive input is a (classical) NP problem. We have seen in the proof of Theorem 1 that NP problems can be decided by testing a single $\mathrm{V\Pi P}^0$ family for zero. Let us now fill in the details.
Digital nondeterminism. Let $L \in \mathbb{NP}_{(K,+,-,=)}$. By [10, Theorem 2], digital nondeterminism suffices : there exists a language $A \in \mathbb{P}_{(K,+,-,=)}$ and a polynomial $p(n)$ such that

$$\bar{x} \in L \iff \exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(\bar{x}, \bar{y}) \in A.$$

Let (C_n) be a family of algebraic circuits of polynomial size $r(n)$ over the structure $(K, +, -, =)$ (i.e. without multiplication gates) that decides A. Notice that our definitions in Valiant's model are constant-free, whereas our algebraic decision circuits (and in particular C_n) may use arbitrary constants. This is not a serious problem : it is enough to consider the constants as new variables (i.e. we pretend that they are part of the input $\bar{x}$), and the circuit is now constant-free. Then our construction leads to a new circuit with the same input variables (and $\mathrm{V\Pi P}^0$ tests). It just remains to plug the original constants in place of the freshly created variables to recognize the original language L. In the remainder of the proof, we therefore assume that the circuits C_n are constant free.

Definition of the arrangement of hyperplanes. Since only addition and subtraction are allowed, on input $(\bar{x}, \bar{y})$ every test in C_n is of the form $\sum_{i=1}^{n} \lambda_i x_i = \sum_{i=1}^{p(|\bar{x}|)} \mu_i y_i + \gamma$, where λ_i, μ_i and γ are integers, and are bounded in absolute value by $2^{r(n)}$. Since $y_i \in \{0,1\}$, the right-hand side of the test is bounded in absolute value by $2^{r(n)}(1 + p(|\bar{x}|))$. Let $q(n)$ be a polynomial satisfying $2^{q(n)} \geq 2^{r(n)}(1 + p(n))$. Consider the family of arrangements $\mathscr{H}_q$ defined in section 5.1 : two points $\bar{x}$ and $\bar{x}'$ in the same cell satisfy

$$\forall \bar{y} \in \{0,1\}^{p(|\bar{x}|)} \; [(\bar{x}, \bar{y}) \in A \iff (\bar{x}', \bar{y}) \in A].$$

Hence these two points both belong to L, or both belong to its complement.

Finding the cell of $\bar{x}$. We can apply Proposition 1 : there is a family of polynomial size algebraic circuits with $\mathrm{V\Pi P}^0$ tests that output the indices of m independent hyperplanes characterizing the cell of $\bar{x}$.

Finding a small rational point in the cell. It is shown in Lemma 5 that we can obtain in polynomial time a point $\bar{q}$ in the cell of $\bar{x}$ of rational coordinates of polynomial size. As pointed out above, $\bar{x}$ is in L if and only if $\bar{q}$ is in L.

Deciding whether a given rational point belongs to L is a problem in NP. It follows from the proof of Theorem 1 that we can decide whether $\bar{q} \in L$ with one additional $\mathrm{V\Pi P}^0$ test. □

Finally, we thank the anonymous referees of [11] for the following remarks.

Remark 3 *The $\mathbb{P}_{(K,+,-,\times,=)}$ algorithm of Theorem 3 in fact does not use arithmetic operations (apart from $\mathrm{V\Pi P}^0$ tests of course). Hence the stronger result $\mathbb{NP}_{(K,+,-,=)} \subseteq \mathrm{P}_{(K,=)}(\mathrm{V\Pi P}^0)$ holds. This does not improve Theorem 2, however.*

Remark 4 *Since $\mathrm{V\Pi P}^0$ can simulate NP (Theorem 1), the inclusion $\mathrm{NP}_{\mathbb{R}_{ovs}} \subseteq \mathrm{P}_{\mathbb{R}_{ovs}}(\mathrm{NP})$ of [6] for $\mathbb{R}_{ovs} = (\mathbb{R}, +, -, \leq)$ implies $\mathrm{NP}_{\mathbb{R}_{ovs}} \subseteq \mathrm{P}_{\mathbb{R}_{ovs}}(\mathrm{V\Pi P}^0)$.*

Références

[1] L. M. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE symposium on foundations of computer science*, pages 75–83, October 1978.

[2] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer-Verlag, 1998.

[3] 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.

[4] P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory.* Number 7 in Algorithms and Computation in Mathematics. Springer, 2000.

[5] H. Fournier and P. Koiran. Are lower bounds easier over the reals ? In *Proc. 30th ACM Symposium on Theory of Computing*, pages 507–513, 1998.

[6] H. Fournier and P. Koiran. Lower bounds are not easier over the reals : Inside PH. In *Proc. ICALP 2000, LNCS 1853*, 2000.

[7] J. Heintz and J. Morgenstern. On the intrinsic complexity of elimination theory. *Journal of Complexity*, 9 :471–498, 1993.

[8] R. M. Karp and R. J. Lipton. Turing machines that take advice. *L'enseignement mathématique*, 28 :191–209, 1982.

[9] P. Koiran. Valiant's model and the cost of computing integers. *Computational Complexity*, 13 :131–146, 2004.

[10] P. Koiran. Computing over the reals with addition and order. *Theoretical Computer Science*, 133(1) :35–48, 1994.

[11] P. Koiran and S. Perifel. Valiant's model : from exponential sums to exponential products. In *Proc. 31th Mathematical Foundations of Computer Science*, 2006, to appear.

[12] R. J. Lipton. Straight-line complexity and integer factorization. In *Proc. First International Symposium on Algorithmic Number Theory*, volume 877 of *Lecture Notes in Computer Science*, pages 71–79. Springer, 1994.

[13] G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard Lyon 1, July 2003.

[14] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[15] B. Poizat. *Les petits cailloux*. Aléas, 1995.

[16] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4) :701–717, October 1980.

[17] M. Shub and S. Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of "NP $\neq$ P ?". *Duke Math. Journal*, 81(1) :47–54, 1995.

[18] L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.