Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Valiant's Model and the Cost of Computing Integers

Pascal Koiran

January 2004

Research Report N° 2004-01

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`

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

INRIA

Valiant's Model and the Cost of Computing Integers

Pascal Koiran

January 2004

Abstract

Let $\tau(k)$ be the minimum number of arithmetic operations required to build the integer $k \in \mathbb{N}$ from the constants 1 and 2. A sequence x_k is said to be "easy to compute" if there exists a polynomial p such that $\tau(x_k) \leq p(\log k)$ for all $k \geq 1$. It is natural to conjecture that sequences such as $\lfloor 2^n \ln 2 \rfloor$ or $n!$ are not easy to compute. In this paper we show that a proof of this conjecture for the first sequence would imply a superpolynomial lower bound for the arithmetic circuit size of the permanent polynomial. For the second sequence, a proof would imply a superpolynomial lower bound for the permanent or P $\neq$ PSPACE.

Keywords: Valiant's model, permanent, factorials, arithmetic circuits.

Résumé

Soit $\tau(k)$ le nombre minimum d'opérations arithmétiques nécessaires pour construire l'entier k à partir des constantes 1 et 2. Une suite d'entiers x_k est dite "facilement calculable" s'il existe un polynôme p tel que $\tau(x_k) \leq p(\log k)$ pour tout $k \geq 1$. Il semble naturel de conjecturer que des suites telles que $\lfloor 2^n \ln 2 \rfloor$ ou $n!$ ne sont pas facilement calculables. Dans cet article nous montrons qu'une preuve de cette conjecture pour la première de ces suites impliquerait une borne inférieure superpolynomiale pour la complexité arithmétique du permanent. Pour la seconde de ces suites, une preuve impliquerait une une borne inférieure superpolynomiale pour le permanent ou P $\neq$ PSPACE.

Mots-clés: modèle de Valiant, permanent, factorielle, circuits arithmétiques.

Valiant's Model and the Cost of Computing Integers

Pascal Koiran

6th January 2004

Abstract

Let $\tau(k)$ be the minimum number of arithmetic operations required to build the integer $k \in \mathbb{N}$ from the constants 1 and 2. A sequence x_k is said to be "easy to compute" if there exists a polynomial p such that $\tau(x_k) \leq p(\log k)$ for all $k \geq 1$. It is natural to conjecture that sequences such as $\lfloor 2^n \ln 2 \rfloor$ or $n!$ are not easy to compute. In this paper we show that a proof of this conjecture for the first sequence would imply a superpolynomial lower bound for the arithmetic circuit size of the permanent polynomial. For the second sequence, a proof would imply a superpolynomial lower bound for the permanent or P $\neq$ PSPACE.

Keywords: Valiant's model, permanent, factorials, arithmetic circuits.

1 Introduction

Let $\tau(k)$ be the minimum number of arithmetic operations ($+$, $-$ or $\times$) required to build the integer $k \in \mathbb{N}$ from the constants 1 and 2. For instance, $\tau(2^{2^k}) = k$ by "repeated squaring." A sequence x_k of integers is said to be "easy to compute" if there exists a polynomial p such that $\tau(x_k) \leq p(\log k)$ for all $k \geq 1$ (one can show for example that 2^k is easy to compute [5]). Otherwise the sequence is said to be "hard to compute". The sequence is said to be "ultimately easy to compute" if there exists another sequence $a_k \in \mathbb{N}$ such that the sequence $a_k x_k$ is easy to compute. It is natural to conjecture that $k!$ is not ultimately easy to compute. Shub and Smale have shown that if this conjecture holds true then P $\neq$ NP over the field of complex numbers [9, 2]. Unfortunately, the conjecture is still open and it is not even known that $k!$ is hard to compute. It is very easy to come up with other examples of sequences which seem hard to compute. For instance, it is tempting to conjecture that the sequences $\lfloor 2^n \ln 2 \rfloor$, $\lfloor 2^n \pi \rfloor$, $\lfloor 2^n e \rfloor$, $\lfloor 2^n \sqrt{2} \rfloor$ and $\lfloor (3/2)^n \rfloor$ are all hard to compute, but proofs seem to be elusive.

It was shown in [5] that for every $\epsilon > 0$, almost all integers satisfy the property $\tau(n) \geq (\log n)/(\log(\log n))^{1+\epsilon}$. The improved lower bound $\tau(n) \geq (\log n)/(\log(\log n))$, which holds again for almost all integers, was established in [8]. These bounds are reminiscent of Shannon's lower bound in boolean complexity theory (see e.g. [11] for a textbook exposition). We conclude that most integers have a high τ complexity, but proving good lower bounds for specific sequences seems to be quite difficult. This situation is again reminiscent of computational complexity theory. In this paper we argue that for some sequences, proving good lower bounds on τ is difficult because they would lead to the solution of major open problems in complexity theory (for instance to a superpolynomial lower bound for the circuit size of the permanent polynomial).

Main results

A quarter of century ago, Valiant proposed an algebraic version of the P versus NP problem [10]. His well-known completeness result for the permanent implies that the class VNP of "easily definable" families of polynomials is equal to the class VP of "easily computable" families if and only if the permanent family is in VP, i.e., can be computed by polynomial size arithmetic circuits. In this paper we establish relations between Valiant's model and the cost of computing integers. The basic idea is quite simple: if an integer polynomial can be evaluated efficiently, its values at integer points are integers of low cost. One difficulty is that in Valiant's model circuits may use arbitrary constants from the underlying field, but we are interested in computing integers "from scratch". It is therefore natural to work with a constant-free version of Valiant's theory. Fortunately, such a theory has recently been studied by Malod in his PhD thesis [7] (see section 2 for a quick introduction). The first relations between Valiant's model and the cost of computing integers are established in section 3. For instance, we show in Theorem 3 that there exists a polynomial p such that $\tau(\lfloor 2^{2^n} \ln 2 \rfloor) \leq p(n)$ for all n under the assumption $\mathrm{VP}^0 = \mathrm{VNP}^0$ (the subscript 0 is used to denote constant-free classes). By the completeness property for the family HC of Hamilton cycle polynomials, this assumption holds true if and only if HC is in VP^0. In section 4 we show that the same results holds true under the (presumably) weaker assumption Permanent $\in \mathrm{VP}^0$. In a very different direction (derandomization of algebraic algorithms), we note that some consequences of the hypothesis that the permanent can be computed by arithmetic circuits of polynomial size have been studied recently in [6]. We show in Theorem 5 that $k!$ is ultimately easy to compute if $\mathrm{VP}^0 = \mathrm{VNP}^0$ and P = PSPACE. The conjuction of these two equalities is an extremely strong assumption, but a refutation seems to be currently out of reach (more on this in section 5). Finally, we give in section 6 a "generalized Valiant criterion" which makes it possible to obtain polylogarithmic bounds on the τ function. Namely, we show that $\lfloor 2^n \ln 2 \rfloor$ is easy to compute if the permanent is in VP^0, and with the additional assumption that P = PSPACE we show that $n!$ is easy to compute.

2 Preliminaries

2.1 Integer Computations

A computation of length l of an integer n is a sequence $(n_{-1}, n_0, n_1, \ldots, n_l)$ of integers such that $n_{-1} = 1$, $n_0 = 2$, $n = n_l$ and for each $i \geq 2$ there exists $j, k < l$ and $\circ \in \{+, -, \times\}$ such that $n_i = n_j \circ n_k$. One sets $\tau(0) = \tau(1) = \tau(2) = 0$ and for $n \geq 3$, $\tau(n)$ is by definition [5] equal to the length of a shortest computation of n. In [2] the number 2 is not allowed as a "starting number", but the two corresponding complexity measures differ by at most 1 since 2 can be obtained from 1 in one arithmetic operation. Some basic properties of the τ function can found in [5]. For instance, $\log\log k \le \tau(k) \le 2\log k$ for any k (use the binary expansion of k for the second inequality); $\tau(2^{2^k}) = k$ by repeated squaring; and $\tau(2^k) \le 2\log k$. The sequence 2^k is therefore easy to compute. The sequence 2^{2^k} is hard to compute for a trivial reason (it grows too quickly as k increases). It seems very plausible that $k!$ is not easy to compute: if it is then "factoring is easy" (see for instance [2], p.126 and [4]). We are not aware of any nontrivial lower bound on $\tau(n!)$. A nontrivial *upper* bound on the arithmetic complexity of computing certain multiples of $n!$ has been published recently in [4]. This upper bound falls short of showing that $n!$ is ultimately easy to compute, however.

2.2 Valiant's Theory without Constants

Valiant's complexity classes are defined relatively to a given field K. Throughout the paper we will take $K = \mathbb{Q}$. We first recall the notion of an *arithmetic circuit*. In such a circuit all gates except the input gates have fan-in 2, and are labelled by $+$, $\times$, or $-$. The input gates are labeled by variables from the set $\{X_1, X_2, \ldots, X_n, \ldots\}$ or by constants from K. If all these constants belong to the set $\{-1, 0, 1\}$, the circuit is said to be *constant-free*. We will assume that there is a single output gate, so that the circuit computes a polynomial in the input variables defined in the usual way. We also define by induction the notion of *formal degree*[1]. The formal degree of an input gate is equal to 1. The formal degree of an addition or subtraction gate is the maximum of the formal degrees of its two incoming gates, and the formal degree of a multiplication gate is the sum of these two formal degrees. Finally, the formal degree of a circuit is equal to the formal degree of its output gate. This is obviously an upper bound on the degree of the polynomial computed by the circuit.

Definition 1 (Malod) *A sequence (f_n) of polynomials belongs to* VP^0 *if there exists a polynomial $p(n)$ and a sequence (C_n) of constant-free arithmetic circuits such that C_n computes f_n and is of size (number of gates) and formal degree at most $p(n)$.*

The size constraint implies in particular that f_n depends on polynomially many variables. The traditional ("non-constant-free") class VP is easily defined in terms of VP^0. Indeed, one can show that a sequence (g_n) of polynomials is in VP iff there exists a sequence (f_n) in VP^0 such that g_n is obtained from f_n by replacing some of the variables by constants from K.

VNP^0 is the other important class in the constant-free theory. It is defined from VP^0 in the natural way.

Definition 2 *A sequence $(f_n(X_1, \ldots, X_{u(n)}))$ belongs to* VNP^0 *if there exists a sequence $(g_n(X_1, \ldots, X_{v(n)}))$ in* VP^0 *such that:*

$$f_n(X_1, \ldots, X_{u(n)}) = \sum_{\overline{\epsilon} \in \{0,1\}^{v(n)-u(n)}} g_n(X_1, \ldots, X_{u(n)}, \overline{\epsilon}).$$

[1] The formal degree is called *degré formel complet* in [7].

Next we give a criterion which makes it easy to recognize many VNP^0 families of polynomials. This result basically goes back to ([10], Remark 1).

Theorem 1 (Valiant's criterion) *Suppose that $n \mapsto p(n)$ is a polynomially bounded function, and that $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is such that the map $1^n 0 j \mapsto f(j,n)$ is in the complexity class $\sharp\mathrm{P}/\mathrm{poly}$. Then the family (f_n) of polynomials defined by*

$$f_n(X_1, \ldots, X_{p(n)}) = \sum_{j \in \{0,1\}^{p(n)}} f(j,n) X_1^{j_1} \cdots X_{p(n)}^{j_{p(n)}} \tag{1}$$

is in VNP^0.

Note that we use a unary encoding for n but a binary encoding for j (j_k denotes the bit of j of weight 2^{k-1}). In the usual statement of this criterion the conclusion is that the family (f_n) is in VNP rather than VNP^0. However, an inspection of the proof (e.g., Proposition 2.20 of [3]) shows that the corresponding construction is constant-free, so that (f_n) is indeed in the smaller class VNP^0. Note also that Theorem 1 covers more families (f_n) than Proposition 2.20 of [3], which only deals with the case where f depends only on its first argument and $p(n) = n$. The proof is essentially unchanged, however. Note also that Theorem 1 is a consequence of Theorem 6 of section 6.

Recall that the Hamilton cycle polynomial HC_n is a function of n^2 variables x_{ij} and is defined by the formula:

$$\mathrm{HC}_n = \sum_{\sigma} \prod_{i=1}^{n} x_{i\sigma(i)}$$

where the sum ranges over all cycles σ of the symmetric group S_n. If $X = (x_{ij})$ is the adjacency matrix of a directed graph G, this polynomial counts the number of Hamilton cycles in G. The following result from [7] gives a "concrete" consequence of the hypothesis $\mathrm{VP}^0 = \mathrm{VNP}^0$.

Theorem 2 $\mathrm{VP}^0 = \mathrm{VNP}^0$ *iff the Hamilton family* (HC_n) *is in* VP^0.

Proof Sketch. The Hamilton family is in VNP^0 by Theorem 1 (the corresponding function ϕ is polynomial-time computable). It is therefore in VP^0 if $\mathrm{VP}^0 = \mathrm{VNP}^0$. The converse follows from the completeness property of (HC_n): any family (f_n) of VNP^0 can be expressed as a projection

$$f_n = HC_{p(n)}(y_1, \ldots, y_{p(n)^2})$$

where $p(n)$ is polynomially bounded and the y_i are either variables or constants from the set $\{-1, 0, 1\}$ [7]. Hence (f_n) is in VP^0 if (HC_n) is in VP^0.
□

In this theorem we use Hamilton polynomials rather than permanents because the completeness proof for the permanent uses divisions by 2 (this is exactly the reason why its completeness proof fails in characteristic 2). It is nonetheless possible to give a somewhat weaker result for the permanent: see Theorem 4 in section 4.

3 An Algebraic Hypothesis

In this section we explore some consequences for the cost of computing integers of the hypothesis $\mathrm{VP}^0 = \mathrm{VNP}^0$.

Proposition 1 *Let (a_n) be an integer sequence such that for some integer b and some polynomially bounded function $p(n)$ one can write:*

$$a_n = \sum_{j=0}^{2^{p(n)}-1} f(j,n) b^j \tag{2}$$

where the map $1^n 0 j \mapsto f(j,n)$ is in $\sharp\mathrm{P}/\mathrm{poly}$. If $\mathrm{VP}^0 = \mathrm{VNP}^0$ then $\tau(a_n)$ is polynomially bounded.

Proof. Consider the family of polynomials

$$g_n(X_1, \ldots, X_{p(n)}) = \sum_{j \in \{0,1\}^{p(n)}} f(j,n) X_1^{j_1} \cdots X_{p(n)}^{j_{p(n)}}.$$

This is a VNP^0 family by Theorem 1. This family is therefore VP^0 under the the assumption $\mathrm{VP}^0 = \mathrm{VNP}^0$, and the result follows from the observation that $a_n = g_n(x_1, \ldots, x_{p(n)})$ where $x_i = b^{2^{i-1}}$. $\square$

Here is an immediate application.

Corollary 1 *Let $a_n = \sum_{k=1}^{2^n} 2^{k^2-1}$. If $\mathrm{VP}^0 = \mathrm{VNP}^0$ then $\tau(a_n)$ is polynomially bounded.*

Proof. Set $b = 2$ and $p(n) = 2n$. Let $f(j,n)$ be the bit of a_n of weight 2^j: $f(j,n) = 1$ if and only if $j \leq 2^{2n} - 1$ and j is of the form $k^2 - 1$. This function is polynomial-time computable, so it is in $\sharp\mathrm{P}$. $\square$

The applications that follow are a little more involved.

Lemma 1 *There is a polynomial time algorithm which takes as inputs three integers k, u and j ($j \leq u$) and computes the bit of $\lfloor 2^u/k \rfloor$ of weight 2^j.*

Proof. Let $N = \lfloor 2^u/k \rfloor$. The difficulty is of course that the bit size of N may be exponential in the input size, so we cannot afford to compute all the bits of N.

We are looking for the bit of weight 2^{-1} of $2^s/k$, where $s = u - j - 1$. This is also the bit of same weight of r/k, where r is the remainder of the Euclidean division of 2^s by k. We are therefore done if r can be computed in polynomial time. For this we use the fact that $\tau(2^s) \leq 2 \log s$ [5] and we perform modulo k all the arithmetic operations in the corresponding computation of 2^s. $\square$

Theorem 3 *Let $l_n = \lfloor 2^{2^n} \ln 2 \rfloor$. If $\mathrm{VP}^0 = \mathrm{VNP}^0$, $\tau(l_n)$ is polynomially bounded.*

Proof. We start from the formula $\ln 2 = \sum_{k=1}^{+\infty} \left(\frac{1}{2}\right)^k / k$,
which implies

$$\sum_{k=1}^{2^n} 2^{2^n-k}/k \leq 2^{2^n} \ln 2 \leq 1 + \sum_{k=1}^{2^n} 2^{2^n-k}/k.$$

It follows that $a_n - 1 \leq l_n \leq a_n + 2^n + 1$, where $a_n = \sum_{k=1}^{2^n} \lfloor 2^{2^n-k}/k \rfloor$. A polynomial bound for $\tau(l_n)$ would therefore follow from a polynomial bound for $\tau(a_n)$. Let $f(j,n)$ be the number of indices $k \in \{1, \ldots, 2^n\}$ such that the bit of weight 2^j in the radix-2 expansion of $\lfloor 2^{2^n-k}/k \rfloor$ is equal to 1. By Lemma 1, the map $(j, 1^n) \mapsto f(j,n)$ is in $\sharp$P. We can therefore put a_n under form (2) with $b = 2$ and $p(n) = n$. The result then follows from Proposition 1. $\square$

One can obtain the same result for several other sequences. For instance, to deal with the sequence $\lfloor 2^{2^n} \ln 3 \rfloor$, observe that $\ln 3 = 2 \ln 2 + \ln(3/4)$ where $\ln(3/4) = -\sum_{k=1}^{+\infty} \left(\frac{1}{4}\right)^k / k$. Similar results can be obtained for expansions in radix different from 2. For instance, to deal with the sequence $\lfloor 3^{2^n} \ln(3/2) \rfloor$, observe that $\ln(3/2) = \sum_{k=1}^{+\infty} \left(\frac{1}{3}\right)^k / k$. In order to apply Proposition 1 with $b = 3$ we then use a version of Lemma 1 where the radix-3 digits of $\lfloor 3^u/k \rfloor$ are computed in polynomial time. More surprisingly, our technique can also be applied to the sequence $\lfloor 2^{2^n} \pi \rfloor$. This follows from the beautiful Bailey-Borwein-Plouffe formula [1]:

$$\pi = \sum_{i=0}^{+\infty} \frac{1}{16^i} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right).$$

For sequences such as $\lfloor 2^{2^n} e \rfloor$, $\lfloor 2^{2^n} \sqrt{2} \rfloor$ or $\lfloor (3/2)^{2^n} \rfloor$ we do not know whether a polynomial complexity bound can be established under the hypothesis $\mathrm{VP}^0 = \mathrm{VNP}^0$.

4 Hamiltonian versus Permanent

We denote by Per_n the $n \times n$ permanent

$$\mathrm{Per}_n = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i\sigma(i)}.$$

The results of the previous section rely on the hypothesis $\mathrm{VP}^0 = \mathrm{VNP}^0$, which by Theorem 2 is equivalent to the hypothesis that the Hamilton family is in VP^0. This hypothesis implies that the permanent family is in VP^0, since it is in VNP^0. The goal of this section is to show that the weaker hypothesis Permanent $\in \mathrm{VP}^0$ implies the same results. We just need to adapt Proposition 1 as follows.

Proposition 2 *Let (a_n) be an integer sequence such that for some integer b and some polynomially bounded function $p(n)$ one can write:*

$$a_n = \sum_{j=0}^{2^{p(n)}-1} f(j,n) b^j \tag{3}$$

where the map $1^n0j \mapsto f(j,n)$ is in $\sharp$P/poly. If the permanent family is in VP^0 *then $\tau(a_n)$ is polynomially bounded.*

The remainder of section 3 is unchanged. For instance, to obtain the counterpart of Theorem 3 one just has to invoke Proposition 2 instead of Proposition 1. The proof of Proposition 2 relies on one theorem and one lemma.

Theorem 4 *Assume that the permanent family is in* VP^0. *For every family (f_n) in* VNP^0, *there exists a polynomially bounded function $p(n)$ such that the family $(2^{p(n)}f_n)$ is in* VP^0.

Proof. By the completeness property of the permanent, any family (f_n) of VNP^0 can be expressed as a projection

$$f_n = \mathrm{Per}_{q(n)}(y_1, \ldots, y_{q(n)^2})$$

where $q(n)$ is polynomially bounded and the y_i are either variables or constants from $\mathbb{Q}$. An inspection of the completeness proof (see for instance [3]) reveals that the constants may all be taken from the set $\{-1, -1/2, 0, 1/2, 1\}$. Assuming that the permanent family is in VP^0, we can therefore write $f_n = C_n(y_1, \ldots, y_{q(n)^2})$ where C_n is a constant-free circuit of size and formal degree bounded by a polynomial function of n. We will now construct a circuit D_n which computes $2^{p(n)}f_n$, where $p(n)$ is the formal degree of C_n. In order to construct D_n from C_n, we replace each gate g of C_n by a subcircuit C_g which will output $2^d g$, where d is the formal degree of g. This construction goes by induction, starting from the input gates. For such a gate the formal degree is equal to 1 by definition, so that C_g needs to output $2x_{ij}$ if g is labeled by the variable x_{ij}, or a constant from the set $\{-2, -1, 0, 1, 2\}$ if g is labeled by a constant. Assume now that g is a computation gate with inputs g_1 of formal degree d_1, and g_2 of formal degree d_2. We first consider the case where g is a multiplication gate. In this case C_g is made of a single multiplication gate with inputs from C_{g_1} and C_{g_2} since $(2^{d_1}g_1)(2^{d_2}g_2) = 2^d g$. If g is an addition gate, $d = \max(d_1, d_2)$. Assume for instance that $d = d_2$. Then C_g needs to output $2^{d_2-d_1}C_{g_1} + C_{g_2}$. Assuming that we have have already computed all the powers of 2 up to $2^{p(n)}$, we only need one addition and one multiplication gate. The case of subtraction gates is similar. The resulting circuit D_n is of polynomial size, and one shows easily by induction that the formal degree of the output gate of C_g is equal to the formal degree of g (for a multiplication gate, use the fact that the gate which outputs $2^{d_2-d_1}$ is of formal degree $d_2 - d_1$). We have therefore shown that the family $(2^{p(n)}f_n)$ is in VP^0. $\square$

Lemma 2 *The inequality $\tau(u) \leq (2\log v + 3)\tau(uv)$ holds true for any pair of integers $u, v \geq 1$.*

Proof. Let $w = uv$ and let $(1, 2, x_1, \ldots, x_l)$ be a computation of w (hence $x_l = w$). We will explain how to compute the sequence (q_i) of the quotients of the euclidean division of x_i by v. Let r_i be the remainder of this division. Since $x_0 = 2$, we have $q_0 \leq 2$ and $r_0 \leq 2$. For $i \geq 1$, we have $x_i = x_j \circ x_k$ where $j, k < i$ and $\circ \in \{+, -, \times\}$.

Consider first the case $\circ = +$. We have $q_i = q_j + q_k$ if $r_j + r_k < v$ and $q_i = q_j + q_k + 1$ otherwise. If $\circ = -$, q_i is either equal to $q_j - q_k$ or to $q_j - q_k - 1$. In both cases, we need at most 2 arithmetic operations to compute q_i from the preceding quotients.

If $\circ = \times$, $x_i = x_j x_k = p_i v + r_j r_k$ where $p_i = q_j q_k v + q_j r_k + q_k r_j$, and $q_i = p_i + \lfloor r_j r_k / v \rfloor$. Since $\lfloor r_j r_k / v \rfloor < v$, $\lfloor r_j r_k / v \rfloor$ can be computed from scratch in at most $2 \log v$ arithmetic operations. In this case q_i can be computed from the preceding quotients in at most $2 \log v + 3$ arithmetic operations. This completes the proof since $u = q_l$. $\square$

Proof of Proposition 2. Consider again the family of polynomials

$$g_n(X_1, \ldots, X_{p(n)}) = \sum_{j \in \{0,1\}^{p(n)}} f(j,n) X_1^{j_1} \cdots X_{p(n)}^{j_{p(n)}}.$$

We have seen in the proof of Proposition 1 that this is a VNP^0 family. Assume that the permanent is in VP^0. By Theorem 4, there exists a polynomially bounded function q such that the family $(2^{q(n)} g_n)$ is in VP^0. Since $a_n = g_n(x_1, \ldots, x_{p(n)})$ where $x_i = b^{2^{i-1}}$, $\tau(2^{q(n)} a_n)$ is polynomially bounded. Now apply Lemma 2 with $u = a_n$ and $v = 2^{q(n)}$. $\square$

It is probably possible to obtain the same results under even weaker hypotheses than Permanent $\in \mathrm{VP}^0$. For instance, one might try to allow rational constants of controlled bit size in arithmetic circuits for the permanent.

5 Boolean and Algebraic Hypotheses

The results of the two previous sections were obtained under hypotheses from algebraic complexity theory ($\mathrm{VP}^0 = \mathrm{VNP}^0$, or Permanent $\in \mathrm{VP}^0$). In this section we show that $n!$ is ultimately easy to compute by adding an hypothesis from boolean complexity theory.

Theorem 5 *If* $\mathrm{VP}^0 = \mathrm{VNP}^0$ *and* $\mathrm{P} = \mathrm{PSPACE}$ *the sequence* $k!$ *is ultimately easy to compute, and in fact* $(2^n)!$ *has polynomially bounded complexity.*

Proof. Let $a_n = (2^n)!$. If $\tau(a_n) \leq q(n)$ for some polynomial q, it is clear that $k!$ is ultimately easy to compute: given k let 2^n be the smallest power of 2 greater or equal to k. Then a_n is a multiple of $k!$, and $\tau(a_n) \leq q(n) \leq q(\log k)$.

Let us therefore assume that $\mathrm{VP}^0 = \mathrm{VNP}^0$ and $\mathrm{P} = \mathrm{PSPACE}$. It remains to show that a_n has polynomially bounded complexity. We would like to apply Proposition 1 with $f(j,n)$ equal to the bit of a_n of weight 2^j, just as in Corollary 1. In order to do this it suffices to show that the map $1^n 0 j \mapsto f(j,n)$ can be computed in polynomial time, or even in polynomial space since $\mathrm{P} = \mathrm{PSPACE}$. We sketch below a parallel algorithm for computing a_n in time polynomial in n (with exponentially many processors). The required polynomial space bound then follows from the equivalence between space and parallel time (see for instance [11], Corollary 2.33).

The parallel algorithm is quite straightforward. We construct a multiplication tree of depth n where the 2^n leaves are labelled by the integers between 1 and 2^n. Each node is supposed to compute the product of the values computed by its two children. The root, which will contain the final result, can be evaluated in n parallel stages. The size of the numbers involved will grow exponentially, but the whole algorithm still runs in polynomial time because the product of two M-bit numbers can be computed in parallel in time $(\log M)^{O(1)}$ (see for instance [11], Theorem 1.23). $\Box$

This technique can be applied to other sequences, and in particular to the sequence $u_n = \lfloor (3/2)^{2^n} \rfloor$ (note that the bit of u_n of weight 2^j is equal to the bit of 3^{2^n} of weight 2^{j+2^n}).

The hypothesis that $\mathrm{VP}^0 = \mathrm{VNP}^0$ and $\mathrm{P} = \mathrm{PSPACE}$ is extremely strong,[2] but apparently cannot be refuted with the known methods of complexity theory. To understand just how strong this hypothesis is, note that $\mathrm{VP}^0 = \mathrm{VNP}^0$ implies NC/poly = PH/poly. This follows from Theorem 4.5 and Corollary 4.6 in [3]. These results as stated in [3] assume Riemann's hypothesis (it is needed in order to eliminate constants). Here we do not need to assume Riemann's hypothesis since we are already working in a constant-free model. Taking into account the additional hypothesis P = PSPACE, we conclude that NC/poly = PSPACE/poly. Note that if we worked with a uniform version of Valiant's model we would conclude instead that NC = PSPACE, an equality which is in contradiction with the space hierarchy theorem.

6 From polynomial to polylogarithmic bounds

The first result of this section is a "generalized Valiant criterion". This name is justified by Remark 1, which shows that Valiant's criterion as stated in Theorem 1 indeed follows from Theorem 6.

Theorem 6 (generalized Valiant criterion) *Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be such that the map $(j,n) \mapsto f(j,n)$ is in the complexity class* $\sharp$P/poly. *Let*

$$f_n(X_1, \ldots, X_{q(n)}) = \sum_{j=0}^{p(n)} f(j,n) X_1^{j_1} \cdots X_{q(n)}^{j_{q(n)}} \tag{4}$$

where j_i denotes the bit of j of weight 2^{i-1}, $q(n) = 1 + \lfloor \log p(n) \rfloor$ and $p(n) \geq n$ for all n.

There exists a VNP^0 family $(g_r(X_1, \ldots, X_r, N_1, \ldots, N_r, P_1, \ldots, P_r))$ which satisfies the following property: for any n,

$$f_n(X_1, \ldots, X_{q(n)}) = g_{q(n)}(X_1, \ldots, X_{q(n)}, n_1, \ldots, n_{q(n)}, p_1, \ldots, p_{q(n)}) \tag{5}$$

where n_i denotes the bit of n of weight 2^{i-1}, and p_i denotes the bit of $p(n)$ of weight 2^{i-1}.

[2] It is clear from the proof that we can replace the hypothesis P = PSPACE by the somewhat weaker hypothesis P/poly = PSPACE/poly.

In contrast to Theorem 1, we use here binary encoding for j *and* n. To be completely precise, we fix the following encoding: a pair (j,n) of integers is represented by a binary string of the form $j_1 \cdots j_r n_1 \cdots n_r$ (i.e, j is represented by the first half of the string, and n by the second half).

Remark 1 *Theorem 1 follows from Theorem 6.*

Proof Sketch. Let (f_n) be a family of polynomials of the form (1). We will assume without loss of generality that $p(n) \geq n+2$ (if not, we can reduce the problem to this situation by adding dummy variables and coding the actual value of $p(n)$ in the advice function). Let $F(j,n) = f(j, \lfloor \log n \rfloor)$. The map $(j,n) \mapsto F(j,n)$ is in $\sharp$P/poly because the map $1^n 0 j \mapsto f(j,n)$ is in $\sharp$P/poly. Let $P(n) = 2^{p(\lfloor \log n \rfloor)} - 1$, and $Q(n) = 1 + \lfloor \log P(n) \rfloor = p\lfloor \log n \rfloor)$. Note that the assumption $p(n) \geq n+2$ implies that $P(n) \geq n$. Finally, let

$$F_n(X_1, \ldots, X_{Q(n)}) = \sum_{j=0}^{P(n)} F(j,n) X_1^{j_1} \cdots X_{Q(n)}^{j_{Q(n)}},$$

and let (G_r) be the VNP^0 family associated to (F_n) by Theorem 6. Since $f_k(X_1, \ldots, X_{p(k)}) = F_{2^k}(X_1, \ldots, X_{p(k)})$, it follows that the family (f_k) is in VNP^0 : f_k appears as a projection of $G_{p(k)}$. $\Box$

Proof of Theorem 6. The assumption that the map $(j,n) \mapsto f(j,n)$ is in $\sharp$P/poly implies that there exists a polynomially bounded function $m(r)$ and a family (p_r) in VP^0 such that for all $j, n \in \{0,1\}^r$,

$$f(j,n) = \sum_{y \in \{0,1\}^{m(r)}} p_r(j,n,y)$$

(here we identify the strings $j, n \in \{0,1\}^r$ with the integers that they represent). This is shown for instance in the proof of Valiant's criterion in [3], whose outline we shall follow. Let X be a tuple of r additional variables, and

$$H_r(X, J, N, Y) = p_r(J, N, Y) \prod_{i=1}^{r} (J_i X_i + 1 - J_i).$$

Note that when $j_1, \ldots, j_r, n_1, \ldots, n_r$ take binary values,

$$f(j,n) X_1^{j_1} \ldots X_r^{j_r} = \sum_{y \in \{0,1\}^{m(r)}} H_r(X, j, n, y).$$

We will also need the existence of a family $(C_r(J_1, \ldots, J_r, P_1, \ldots, P_r))$ in VP^0 such that $C_r(j,p) = 1$ if $j \leq p$, and $C_r(j,p) = 0$ if $j > p$. This can be shown by induction on r, using the fact that for boolean values of the variables and $r > 1$, $C_r(j,p)$ is equivalent to

$$(p_r = 1 \wedge j_r = 0) \vee (p_r = j_r \wedge C_{r-1}(j_1, \ldots, j_{r-1}, p_1, \ldots, p_{r-1})).$$

Then one represents boolean operations by polynomials in the standard way (for instance $u \wedge v$ is represented by UV, and $u \vee v$ by $U+V-UV$).

Let $G_r(X,J,N,Y,P) = C_r(J,P)H_r(X,J,N,Y)$. The family

$$g_r(X,N,P) = \sum_{j\in\{0,1\}^r} \sum_{y\{0,1\}^{m(r)}} G_r(X,J,N,Y,P)$$

is in VNP^0 since G_r is in VP^0. By construction, $g_r(X,n,p) = \sum_{j=0}^{p} f(j,n)X_1^{j_1}\ldots X_r^{j_r}$ and (5) follows immediately by setting $p = p(n)$ and $r = q(n)$ (here we use the assumption $p(n) \geq n$ to ensure that the binary encoding of n fits within r bits). $\Box$

Corollary 2 *Let (f_n) be the family of polynomials defined by (4), and assume additionally that $n \mapsto p(n)$ is a polynomially bounded function.*

If $\mathrm{VP}^0 = \mathrm{VNP}^0$*, (f_n) can be computed by a family of constant-free circuits of size* $(\log n)^{O(1)}$.

If we assume only that the permanent is in VP^0 *then there exists a polylogathmically bounded function $s(n)$ such that $2^{s(n)}f_n$ can be computed by a family of constant-free circuits of size* $(\log n)^{O(1)}$.

Proof. If $\mathrm{VP}^0 = \mathrm{VNP}^0$, the family (g_r) of Theorem 6 is in VP^0, and can thus be computed by a a family of constant-free circuits of size $r^{O(1)}$. In view of (5), we obtain a family of constant-free circuits of size $(\log n)^{O(1)}$ for f_n.

Let us now assume only that the permanent is in VP^0. By Theorem 4 there exists a polynomially bounded function p such that the family $(2^{p(r)}g_r)$ is in VP^0, and the result follows again from (5). $\Box$

Proposition 3 *Suppose that $n \mapsto p(n)$ is a polynomially bounded function, and that $p(n) \geq n$ for all $n \in \mathbb{N}$. Let (a_n) be an integer sequence such that for some integer b one can write:*

$$a_n = \sum_{j=0}^{p(n)} f(j,n)b^j \tag{6}$$

where the map $(j,n) \mapsto f(j,n)$ is in $\sharp\mathrm{P/poly}$. *If the permanent family is in* VP^0 *then (a_n) is easy to compute.*

Proof. It is a variation on the proof of Proposition 2. Let (f_n) be the family of polynomials defined by (4). If the permanent is in VP^0, by Corollary 2 there exists a polylogathmically bounded function $s(n)$ and a family (C_n) of constant-free circuits of size $(\log n)^{O(1)}$ which compute $2^{s(n)}f_n$. Since $a_n = f_n(x_1,\ldots,x_{q(n)})$ where $x_i = b^{2^{i-1}}$, $\tau(2^{s(n)}a_n) = (\log n)^{O(1)}$. Now apply Lemma 2 with $u = a_n$ and $v = 2^{s(n)}$. $\Box$

Finally, we give two results which respectively improve Theorem 3 and Theorem 5.

Theorem 7 *If the permanent is in* VP^0 *the sequence $L_n = \lfloor 2^n \ln 2 \rfloor$ is easy to compute.*

Proof. It is a variation on the proof of Theorem 3. Now we use the fact that

$$\sum_{k=1}^{n} 2^{n-k}/k \leq 2^n \ln 2 \leq 1 + \sum_{k=1}^{n} 2^{n-k}/k.$$

It follows that $A_n - 1 \leq L_n \leq A_n + n + 1$, where $A_n = \sum_{k=1}^{n} \lfloor 2^{n-k}/k \rfloor$. Let $f(j,n)$ be the number of indices $k \in \{1, \ldots, n\}$ such that the bit of weight 2^j in the radix-2 expansion of $\lfloor 2^{n-k}/k \rfloor$ is equal to 1. This is a $\sharp$P function by Lemma 1. We can therefore put A_n under form (6) with $b = 2$ and $p(n) = n$. It follows from Proposition 3 that (A_n) is easy to compute, and the same is therefore true of (L_n). $\square$

Theorem 8 *If the permanent is in* VP^0 *and* P = PSPACE, $n!$ *is easy to compute.*

Proof. By Proposition 3, it suffices to show that the bit of $n!$ of weight 2^j can be computed in space polynomial in the bit size of the pair (j,n). This is done essentially as in the proof of Theorem 5. $\square$

References

[1] D. Bailey, P. Borwein, and S. Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66(218):903–913, 1997.

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

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

[4] Q. Cheng. On the ultimate complexity of factorials. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 157–166. Springer, 2003.

[5] W. De Melo and B. F. Svaiter. The cost of computing integers. *Proc. American Mathematical Society*, 124(5):1377–1378, 1996.

[6] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity test means proving circuit lower bounds. In *Proc. 35th ACM Symposium on Theory of Computing*, pages 355–364, 2002.

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

[8] C. Moreira. On asymptotic estimates for arithmetic cost functions. *Proc. American Mathematical Society*, 125(2):347–353, 1997.

[9] M. Shub and S. Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of "P=NP". *Duke Mathematical Journal*, 81(1):47–54, 1996.

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

[11] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.