Available online at www.sciencedirect.com
ScienceDirect
Journal of Complexity 24 (2008) 16–38
Journal of COMPLEXITY
www.elsevier.com/locate/jco

Characterizing Valiant's algebraic complexity classes

Guillaume Malod[a,*], Natacha Portier[b]

[a] *Kyoto University, Japan*
[b] *École Normale Supérieure de Lyon, France*

Received 30 January 2006; accepted 27 September 2006
Available online 27 April 2007

Abstract

Valiant introduced 20 years ago an algebraic complexity theory to study the complexity of polynomial families. The basic computation model used is the arithmetic circuit, which makes these classes very easy to define and open to combinatorial techniques. In this paper we gather known results and new techniques under a unifying theme, namely the restrictions imposed upon the gates of the circuit, building a hierarchy from formulas to circuits. As a consequence we get simpler proofs for known results such as the equality of the classes VNP and $\mathrm{VNP_e}$ or the completeness of the Determinant for VQP, and new results such as a characterization of the classes VQP and VP (which we can also apply to the Boolean class LOGCFL) or a full answer to a conjecture in Bürgisser's book [Completeness and reduction in algebraic complexity theory, Algorithms and Computation in Mathematics, vol. 7, Springer, Berlin, 2000]. We also show that for circuits of polynomial depth and unbounded size these models all have the same expressive power and can be used to characterize a uniform version of VNP.

MSC: 03D15; 68Q15; 68Q17

Keywords: Algebraic complexity; Valiant's theory; Polynomials; Straight line programs; Circuits; Permanent; Determinant; Skew circuits; Formulas

1. Introduction

The common case in favor of studying arithmetic circuits is that they offer a compact representation of polynomials (the Determinant polynomial for instance has a factorial number of monomials but a polynomial size circuit representation). Results by Kaltofen [16] and von zur

* Corresponding author.
E-mail addresses: malod@kuis.kyoto-u.ac.jp, guillaume.malod@umh.ac.be (G. Malod), Natacha.Portier@ens-lyon.fr (N. Portier).

0885-064X/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jco.2006.09.006

Gathen [32] show that standard symbolic manipulations can be applied to arithmetic circuits, and thus the feasibility of such a representation scheme. Valiant's algebraic complexity classes appear in this context as the relevant formalization of intractability, explaining for instance why we have no efficient algorithm for general iterated derivation, because it would imply the collapse of VP and VNP. Arithmetic circuits are also linked to Boolean complexity. Kabanets and Impagliazzo [15] link the de-randomization of Polynomial Identity Testing with super-polynomial arithmetic circuit lower bounds for the Permanent (i.e. the separation of the classes VP and VNP). Koiran [17] shows that the complexity of computing certain integers such as $n!$ is related to Valiant's classes. Arithmetic circuits can also be considered as Boolean inputs and define new problems with interesting consequences in complexity, as is shown by [2], where the problem of deciding whether an arithmetic circuit computes a positive number is related to numerical analysis.

Our interest in Valiant's classes is based on a slightly different perspective. These classes can be seen as representing computations by circuits in general, be they arithmetic or Boolean. Circuits are often used as alternative characterizations of important classical complexity classes such as P, NP and #P (cf. [29]), or to define new classes (cf. [1]). The definition of Valiant's classes is very simple so that the combinatorial insights are unfettered by computational details. Moreover the reductions used are low level (p-projections, as used for example in [13]), thus giving very strong completeness results.

We stress this combinatorial aspect by introducing restrictions on circuits to give a characterization of the class VP (Theorem 1) and of the class VQP (Proposition 4), the latter class capturing the complexity of the Determinant. The characterization of VP greatly simplifies one the main steps in the completeness proof of the Permanent, namely showing the equivalence of circuits and formulas under a Boolean sum. It also illustrates our point on the general nature of these classes, because we can deduce a new circuit characterization of LOGCFL and $\#\mathrm{SAC}^1$ (Propositions 2 and 3). The characterization of VQP yields a full answer to a conjecture by Bürgisser [7] stating that several operations of linear algebra are VQP-complete (Theorem 5). The techniques used to study the class VQP are in fact similar to those used by Toda [25]. We import his definition of a class capturing the complexity of the Determinant and suggest that it is better suited to the task than VQP. This provides a new non-computational characterization of the question of VP versus VNP as the old problem of computing a Permanent as the Determinant of a matrix of polynomial size (Theorem 8). The completeness of the Determinant also provides a positive answer to another question raised by Bürgisser, namely that the Determinant family is indeed linearly closed (Proposition 7). We finally use similar circuit techniques to characterize a uniform version of VNP (Theorem 10).

2. Basic definitions

We give here a brief introduction to Valiant's theory. Detailed information can be found in [7,32]. Valiant's algebraic classes revolve around the representation of polynomials over a given field by arithmetic circuits. These polynomials are abstract, in the sense that they are defined by the sequence of their coefficients. One should remember to distinguish polynomials in this sense from polynomial functions, which are the functions defined by polynomials over a field.

Definition 1. An *arithmetic circuit* is a finite acyclic directed graph with vertices of in-degree 0 or 2 and exactly one vertex of out-degree 0. Vertices of in-degree 0 are called *inputs* and labeled by a constant or a variable. The other vertices, of in-degree 2, are labeled by $\times$ or $+$ and called

computation gates. We distinguish left and right arguments to a computation gate (i.e. each arrow in our graph is implicitly labelled with L or R depending on whether it is a left or right input). The vertex of out-degree 0 is called the *output*. The vertices of a circuit are commonly called *gates* and its edges *arrows*.

The polynomial represented by a circuit can easily be defined by induction. Circuits represent a computation where one can reuse partial results. If we do not allow this, that is if we require each argument to be computed especially for a given computation step, then the graph underlying the circuit must be a tree. Such circuits are called *expressions*, *arithmetic terms* or *formulas* (we shall use the latter).

Definition 2. The *size* of a circuit is its number of gates. The *depth* is the maximal length of a directed path from an input to an output. The *degree* of a gate is defined recursively: any input is of degree 1; the degree of a $+$ gate is the max of the incoming degrees; the degree of a $\times$ gate is the sum of the incoming degrees. The degree of the circuit is the degree of its output gate.

As usual in complexity theory we are interested in asymptotics, in this case the growth of the size of the circuits representing a sequence of polynomials. We give here the definitions of Valiant's classes and the reductions used. Note that the classes depend on a chosen field, but as we are interested in combinatorial techniques this will almost never play a role in this paper.

Definition 3. A sequence of polynomials (f_n) belongs to VP if there exists a sequence of circuits (C_n) of polynomially bounded size and degree such that C_n represents f_n. A sequence of polynomials (f_n) belongs to VNP if there exists a polynomial p and a sequence $(g_n) \in$ VP such that $f_n(\bar{x}) = \sum_{\bar{\varepsilon}\in\{0,1\}^{p(|\bar{x}|)}} g_n(\bar{x}, \bar{\varepsilon})$.

A polynomial f is a *projection* of a polynomial g if $f(\bar{x}) = g(a_1, \ldots, a_m)$, where the a_i are elements of the field or variables among $x_1, \ldots, x_n$. A sequence (f_n) is a *p-projection* of a sequence (g_n) if there exists a polynomially bounded function $t(n)$ such that f_n is a projection of $g_{t(n)}$ for all n.

It is obvious that VP is included in VNP. Valiant's hypothesis is that this inclusion is strict; it remains a major open problem of complexity theory. The definition of VP given here bounds both the degree and the size of the circuit representing a polynomial. The definition in [7] bounds the degree of the represented polynomial and the size of the circuit. One can show that these definitions are equivalent. The following classes are defined using formulas in place of circuits and play an important part in the completeness proof of the Permanent.

Definition 4. A sequence of polynomials (f_n) belongs to the class $\mathrm{VP_e}$ if there exists a sequence of formulas (F_n) of polynomially bounded size such that F_n represents f_n.
A sequence of polynomials (f_n) belongs to $\mathrm{VNP_e}$ if there exists a polynomial p and a sequence $(g_n) \in \mathrm{VP_e}$ such that $f_n(\bar{x}) = \sum_{\bar{\varepsilon}\in\{0,1\}^{p(|\bar{x}|)}} g_n(\bar{x}, \bar{\varepsilon})$.

The main result in Valiant's theory is the completeness of the Permanent family of polynomials for the class VNP, over fields of characteristic different from 2. The Permanent of a matrix of size n with variables entries $z_{i,j}$ is defined as $\mathrm{PER}_n(z_{i,j}) = \sum_{\sigma\in S_n} \prod_{i=1}^{n} z_{i,\sigma(i)}$. In this definition, S_n is the group of permutations of $\{1, \ldots, n\}$. This result stands in stark contrast to the fact that the Determinant family belongs to the class VP. The Determinant is defined as the Permanent but with

positive and negative monomials depending on the sign $s(\sigma)$ of the permutation: $\mathrm{DET}_n(z_{i,j}) = \sum_{\sigma\in S_n} s(\sigma)\prod_{i=1}^{n} z_{i,\sigma(i)}$.

3. Characterizing VP

Whereas the class VNP captures the complexity of the Permanent and many other problems, there is no natural complete problem for the class VP, which is still not very well understood. We give here an intuitive characterization which we hope may provide better insight. For this purpose we introduce the following definition, exploiting the interplay between circuits and formulas in Valiant's theory.

Definition 5. Let α be a gate receiving arrows from gates β and γ. We say that α is *disjoint* if the sub-circuits associated to β and γ are disjoint from one another. A circuit is *multiplicatively disjoint* (MD) if all its multiplication gates are disjoint.

The circuit in Fig. 1 is MD, as shown by the depiction of its multiplication gates and their respective sub-circuits. One can see MD circuits as intermediate between formulas and circuits. A circuit is a formula if and only if all its gates are disjoint. A MD circuit behaves like a formula for multiplications. Disjoint multiplications can be seen as a way to control the degree of the polynomial computed by a circuit, which links this technique to the retarded multiplication scheme used in [4] to characterize the class #P. However, it also provides combinatorial information which we will use in the next section. Let us now show that MD circuits enable us to characterize VP.

Theorem 1. *A sequence of polynomials* (f_n) *belongs to* VP *if and only if there exists a sequence* (C_n) *of MD circuits, of polynomially bounded size, such that* C_n *represents the polynomial* f_n.

Proof. This theorem is an obvious consequence of Lemmas 1 and 2 below. □

Lemma 1. *If C is a MD circuit of size t, its degree is less than t.*

Lemma 2. *If C is a circuit of size t and degree d, there exists a MD circuit C′, which computes the same polynomial and whose size is less than dt.*

Lemma 1 can be shown by an easy induction on the size of the circuit. The basic idea behind the proof of Lemma 2 is comparable to the naïve transformation of a circuit into a formula by

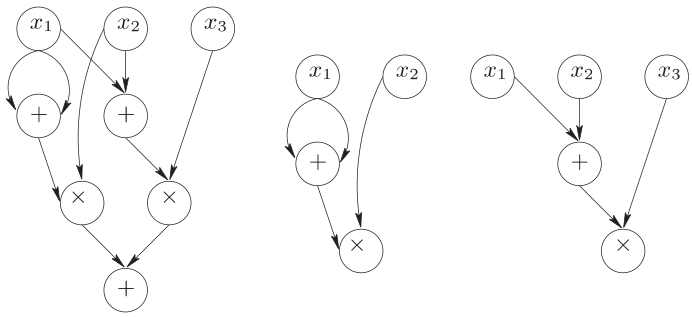

Fig. 1. A multiplicatively disjoint circuit.

duplicating gates, however, the degree of the circuit turns out to be a bound on the number of copies needed for a given gate, so that we avoid a potentially exponential growth in size. We describe here such a direct proof, however, this lemma can also be seen as a consequence of the characterization of circuits of polynomial size and degree by semi-unbounded circuits of polynomial size and logarithmic depth [3].

Proof of Lemma 2. As will happen in several other proofs, we allow circuits to have several output gates. We shall build a sequence of MD circuits C_f, with f ranging from 1 to d such that for any gate α of C which is of degree e less than f the following conditions hold:

- C_f should contain distinct gates $\alpha_1, \ldots, \alpha_{d+1-e}$ which each compute the polynomial computed by α in C; gate α_k is called the clone of α of index k,
- the gates in the sub-circuit of C_f associated with the clone α_k are clones whose index lies between k and $k+e-1$ included.

Circuit C_1 is made of d copies of the sub-circuit of C containing only gates of degree 1. Therefore it does not contain any multiplication gate and is thus MD. Each gate α of C of degree 1 has d clones and the gates of the sub-circuit associated with α_k are clones of index $k = k+1-1$. The aforementioned conditions are met.

Suppose now that the circuits C_f have been built up to $e-1$. We start by adding multiplication gates. Let α be a multiplication gate in C of degree e, receiving arrows from gates β and γ of degree e_1 and e_2, respectively (with $e = e_1 + e_2$). We add the clones $\alpha_1, \ldots, \alpha_{d+1-e}$. For i ranging from 1 to $d+1-e$, α_i receives an arrow from the clone β_i and an arrow from the clone γ_{i+e_1} of C_{e-1} (these clones exist because $1 \leqslant i \leqslant d+1-e$ and $e_1 + 1 \leqslant i + e_1 \leqslant d + 1 - e_2$). Since each clone of β in C_{e-1} computes the same polynomial as β in C, and similarly for γ, each clone of α in C_{e-1} computes the polynomial computed by α in C. In order to show that the resulting circuit is MD, we only need to check that each gate α_i is disjoint. But the gates in the sub-circuit associated with β_i are clones whose index lies between i and $i + e_1 - 1$ and the gates in the sub-circuit associated with γ_{i+e_1-1} are clones whose index lies between $i + e_1$ and $i + e_1 + e_2 - 1$. The two sub-circuits which send an arrow to α_i are therefore disjoint. Finally one can check the last required property: the sub-circuit associated with α_i is the union of the sub-circuit associated with β_i with the sub-circuit associated with γ_{i+e_1-1}. The gates are therefore clones of index ranging from i to $i + e_1 + e_2 - 1 = i + e - 1$.

We then add the addition gates, following an order such that when we clone a gate, each gate from which it receives an arrow has already been cloned. Let α be an addition gate in C of degree e, receiving arrows from gates β and γ of respective degree e and e' (with $e' \leqslant e$). We add the clones $\alpha_1, \ldots, \alpha_{d+1-e}$. For i ranging from 1 to $d+1-e$, α_i receives an arrow from the clone β_i and an arrow from the clone γ_i. Since we are adding an addition gate, the circuit stays MD. Each clone of α computes the adequate polynomial. And the gates of the sub-circuit associated with α_i are clones whose index lies between i and $i + e - 1$, because e' is less than e.

Let C' be the associated sub-circuit for the output gate of C in C_d. By construction this circuit is MD and computes the same polynomial as C. Each gate in C has been cloned at most d times, so the size of C' is less than dt. □

Our characterization of VP uses circuits which seem to be the middle ground between formulas and circuits. It is therefore not so surprising that we should be able to use this characterization to compare the expressive power of both models.

4. Consequences

4.1. Formulas and circuits

One major open question is whether circuits are more powerful than formulas at the polynomial level, i.e. whether the inclusion $\mathrm{VP_e} \subseteq \mathrm{VP}$ is strict or not. The first step of the completeness proof of the Permanent is to show that under a Boolean sum formulas and circuits have the same power. A technically involved proof of this can be found for example in [7]. We use our characterization of VP to give a simpler and more intuitive proof.

Theorem 2. $\mathrm{VNP} = \mathrm{VNP_e}$ *over any field.*

We will introduce some useful definitions and properties before giving the proof. Of the two inclusions in this equality, the harder to show is $\mathrm{VNP} \subseteq \mathrm{VNP_e}$. However, the definition of VNP implies that this inclusion is a consequence of $\mathrm{VP} \subseteq \mathrm{VNP_e}$. We therefore need to express the polynomial represented by a circuit as a sum of formulas. For a given circuit we will consider graphs called *parse trees*. These graphs appear under different names in several previous works [3,14,29,30]. We will use them in the context of arithmetic circuits, in the spirit of this quote from [14]: a parse tree is "a family tree which charts the generation of a particular monomial in the final result".

Definition 6. The set of parse trees of a circuit C is defined by induction on its size:

- If C is of size 1 it has only one parse tree, itself.
- If the output gate of C is a $+$ gate whose arguments are the gates α and β, the parse trees of C are obtained by taking either a parse tree of C_α and the arrow from α to the output or a parse tree of C_β and the arrow from β to the output.
- If the output gate of C is a $\times$ gate whose arguments are the gates α and β, the parse trees of C are obtained by taking a parse tree of C_α and a parse tree of a disjoint copy of C_β and the arrows from α and β to the output.

Recall that arrows are labeled as left and right inputs, and this transfers to parse trees, i.e. when we choose one input arrow for a $+$ gate, it keeps its label. We may also describe parse trees in the following manner. If C is a MD circuit, a graph T is a *parse tree* of C if the following conditions are met:

(1) T is a subgraph of C which contains the output gate of C.
(2) If α is a multiplication gate in T receiving arrows from gates β and γ in C, then the arrows (β, α) and (γ, α) both also appear in T.
(3) If α is an addition gate in T, it receives exactly one arrow in T.
(4) Only arrows and gates obtained in this way belong to T.

Fig. 2 gives an example of a circuit and its parse trees. Each parse tree is identified with a monomial by computing the product of the values of the input gates. It turns out that the polynomial computed by the circuit is thus the sum of the values of its parse trees. This is true in general, and can easily be shown by induction. We write $\mathrm{val}(T)$ for the value of parse tree T.

Lemma 3. *If C represents the polynomial f then $f(\bar{x}) = \sum_T \mathrm{val}(T)$, where the sum is over the set of parse trees of C.*

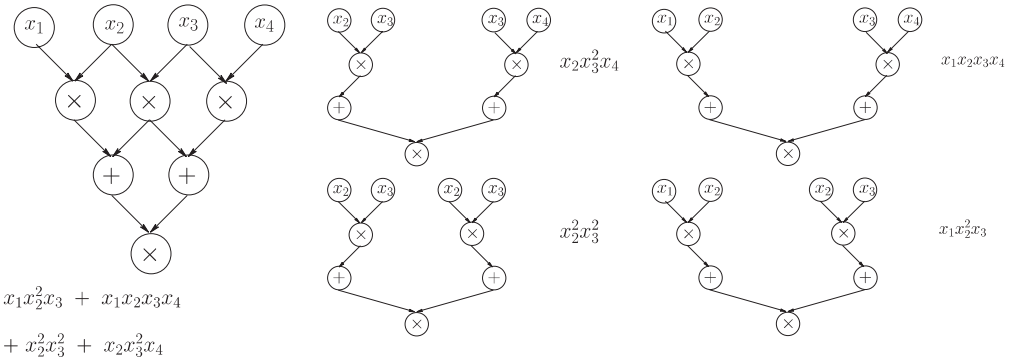

Fig. 2. A circuit and its parse trees.

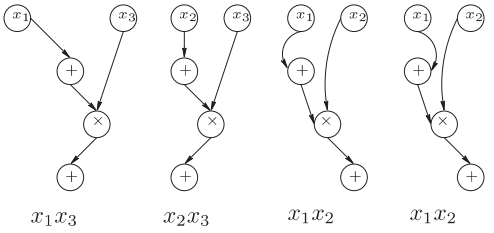

Fig. 3. The parse trees of the circuit from Fig. 1.

To prove Theorem 2, we will need to "recognize" parse trees encoded as Boolean words. This task is easier for MD circuits, thanks to the following proposition, which is not hard to prove.

Proposition 1. *A circuit C is MD iff any parse tree of C is a subgraph* [1] *of C.*

For instance, Fig. 3 gives the parse trees of the circuit from Fig. 1 and one can see that they are subgraphs of the circuit. McKenzie et al. study in [19] the notion of parse trees (which they call proof trees) and the associated notion of proof circuits, defined as sub-circuits satisfying conditions (1)–(4) above. They study the complexity of counting trees versus counting circuits. In their setting, one can see MD circuits as the circuits whose sets of parse trees and proof circuits are equal.

Proof of Theorem 2. The inclusion $\mathrm{VNP_e} \subseteq \mathrm{VNP}$ is obvious. As noted before, for the converse we only need to show that $\mathrm{VP} \subseteq \mathrm{VNP_e}$.

Consider a polynomial sequence in VP. We will use Lemma 3 to express it as a sum of formulas, but we need to show that we can indeed sum over all parse trees and compute the value of a parse tree. In other words we will in fact sum over all possible Boolean words of a given length, as in the definition of $\mathrm{VNP_e}$, therefore we need to have a formula to recognize when a word encodes a parse tree and to compute its value.

[1] Subgraph in the sense of graph theory [11]: a graph $G = (V, E)$ is a subgraph of a graph $G' = (V', E')$ if $V \subseteq V'$ and $E \subseteq E'$.

The useful implication of Proposition 1 is that in the case of MD circuits all parse trees are subgraphs. And since the circuit is of polynomial size, it is straightforward, if somewhat tedious, to recognize and compute the value of the parse trees of a circuit with a formula.

By using an addition with 0 we can modify the circuit to ensure that no addition gate receives both of its arrows from the same gate. Let us label the gates of C with the numbers from 1 to t. We then partition the set $\{1, 2, \ldots, t\}$ in three sets I, M, A which, respectively, contain the labels for input gates, multiplication gates and addition gates and let us suppose that t labels the output gate. For i in E, let V_i be the variable for the input gate i. A parse tree D shall be encoded by the variables $a_{i,j}$ for i and j ranging from 1 to t and such that the arrow (i, j) belongs to C, with the idea that this variable is 1 if the arrow (i, j) is in D and 0 otherwise, and by the variables p_i for i ranging from 1 to t, this variable being 1 if gate i is in D and 0 otherwise. We shall compute the product of the following polynomials, each being used to meet one of the requirements in the definition of a parse tree of a circuit. We start by demanding that if an arrow is in D then the gates it links must belong to D: $\prod_{(i,j)\in C}(a_{i,j}p_i p_j + 1 - a_{i,j})$.

(1) To ensure that D contains the output gate of C: p_t.
(2) To ensure that for any multiplication gates in D, both arrows it receives are also in D:

$$\prod_{\substack{i\in M \text{ and } j,k \text{ such that} \\ (j,i)\in C \text{ and } (k,i)\in C}} (p_i a_{j,i} a_{k,i} + (1 - p_i)).$$

(3) To ensure that for any addition gate in D, it receives exactly one arrow in D:

$$\prod_{\substack{i\in A \text{ and } j,k \text{ such that} \\ (j,i)\in C \text{ and } (k,i)\in C}} (p_i\ (a_{j,i}(1 - a_{k,i}) + a_{k,i}(1 - a_{j,i})) + 1 - p_i\ ).$$

(4) To ensure that any gate in D which is not the output gate sends at least one arrow toward another gate in D:

$$\prod_{1\leqslant i<t} \left(p_i \cdot \left(\sum_{\substack{j \text{ such that} \\ (i,j)\in C}} a_{i,j} \right) + 1 - p_i \right).$$

(Do note that if a subgraph of a MD circuit satisfies conditions (2)–(4) then any of its gates sends at most one arrow.)

After having checked that $\bar{a}$, $\bar{p}$ does encode a parse tree of C, we use the following polynomial to compute the associated monomial: $\prod_{i\in E}(p_i \cdot V_i + 1 - p_i)$. These polynomials can clearly be computed by arithmetic formulas of polynomial size with regard to the number of gates in the MD circuit C. Taking the sum over all Boolean words $\bar{a} \in \{0, 1\}^{t\times t}$ and $\bar{p}\ in \{0, 1\}^t$ expresses the value of our circuit as a sum of formulas, as required. □

4.2. Boolean classes defined by MD circuits

In this section, we illustrate the general nature of results in Valiant's theory by applying them in the Boolean setting. The name VP might suggest that the related Boolean class is P, but if one looks at circuit definitions of Boolean classes, it is obvious that the closest class is LOGCFL.

Definition 7. LOGCFL is the class of decision problems that can be reduced in logarithmic space to a context-free language.

This link is reflected in the *polynomial proof tree size* property identified by Venkateswaran [28] to characterize LOGCFL. In our context, all parse trees of a MD circuit are sub-circuits, and thus have the same size bound as the circuit. We get a straightforward characterization of LOGCFL by MD circuits, where we use the Boolean operators as ring operations and define the degree of a circuit accordingly.

Proposition 2. *LOGCFL is the class of languages accepted by uniform sequences of MD Boolean circuits of polynomial size.*

Proof (hint). The proof uses Venkateswaran's characterization of LOGCFL by semi-unbounded Boolean circuits of polynomial size and logarithmic depth in [28] (which explains why LOGCFL is also sometimes called SAC^1) and the equivalence with MD circuits of polynomial size, which is not hard to show. □

The class $\#\mathrm{SAC}^1$, the counting class associated with LOGCFL, is defined by arithmetizing the circuit definition in the manner described in [1]: replace each $\wedge$ gate by a multiplication gate, and each $\vee$ gate by an addition gate (negated inputs $\neg x$ are replaced by $(1-x)$) and you get the counting class. Its characterization below is then straightforward.

Definition 8. $\#\mathrm{SAC}^1$ is the class of functions from $\{0,1\}$ to $\mathbb{N}$ computed by uniform sequences of arithmetic circuits of polynomial size and degree.

Proposition 3. $\#\mathrm{SAC}^1$ *is the class of functions computed by uniform sequences of MD arithmetic circuits of polynomial size.*

Note that these results can be contrasted to the known links between NC^1 and formulas on one hand, and NL and skew circuits on the other hand. The hierarchy we are studying in this paper also exists in the Boolean case.

5. The complexity of the Determinant

5.1. The class VQP

The Determinant family is known to belong to the class VP. However, it is not known to be VP-complete, nor is it thought to be. The class VQP, defined via circuits of quasi-polynomial size, was introduced to further study the complexity of the Determinant. Indeed one can find proofs of completeness of the Determinant for VQP in [7,32]. We give here a simple proof using a stronger restriction on multiplications than the one used to characterize VP. Note that the notion of reduction is also changed in the definitions below.

Definition 9. A function t from $\mathbb{N}$ to $\mathbb{N}$ is *quasi-polynomially bounded* if there exist two constants a and b such that $t(n) \leqslant n^{a\cdot\log^b n}$ for all $n \geqslant 2$.

A sequence of polynomials (f_n) belongs to the class VQP if its number of variables and degree is polynomially bounded and if it is represented by a circuit of quasi-polynomially bounded size.

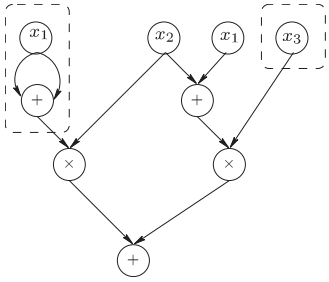

Fig. 4. A weakly skew circuit.

A sequence (f_n) is a *qp-projection* of a sequence (g_n) if there exists a quasi-polynomially bounded function t such that for all n f_n is a projection of $g_{t(n)}$.

The proof given in [7] relies on a parallelization theorem [27] which states that a circuit of size s and degree d in n variables can be parallelized to produce a circuit of size $O(d^6 s^3)$ and depth $O((\log ds)\log d + \log n)$. A stronger version of MD circuits was used in [18] to prove the same completeness result without the need to parallelize. These so-called *strongly MD* circuits are in fact the *weakly skew* circuits of [25]. This last work is extremely relevant to the complexity of the Determinant in Valiant's: the connection between skew or weakly skew circuits and the Determinant of integer matrices has thus already been observed, but it is surprising that the class VQP is used to capture the complexity in Valiant's setting (cf. [7,32]) when a definition based on skew circuits is much more natural.

Much as MD circuits give us more information than the retarded programs of Babai and Fortnow [4], weakly skew circuits provide the necessary structural information when compared to the restricted programs introduced by Damm [9]. Recall that a circuit is *skew* if all multiplication gates have at most one argument which is not an input gate. The condition is somewhat relaxed for weakly skew circuits.

Definition 10. A circuit is *weakly skew* if for any multiplication gate α, receiving arrows from gates β and γ, one of the two sub-circuits C_β or C_γ is only connected to the rest of the circuit by the arrow going to α.

Formulas are circuits where arguments cannot be re-used, weakly skew circuits demand that at least one of the two arguments of a multiplication gate be computed just for that gate. Fig. 4 gives an example of a weakly skew circuit and shows the independent argument of each multiplication gate. Weakly skew circuits characterize VQP.

Proposition 4. *A sequence of polynomials* (f_n) *belongs to* VQP *if and only if there exists a sequence* (C_n) *of weakly skew circuits, of quasi-polynomially bounded size and polynomially bounded degree, such that* C_n *represents the polynomial* f_n.

Proof. A consequence of Lemma 4 below. □

Note that if one defined a bigger class VQP* by only imposing a quasi-polynomial bound on the degree instead of a polynomial bound, then the characterization is exact in the following sense: a

sequence of polynomials belongs to VQP* if and only if it is represented by a sequence of weakly skew circuits of quasi-polynomially bounded size. All the completeness results for VQP as defined above would hold for such a class VQP*. Note also that this result also holds for formulas as can also be shown by using the parallelization theorem quoted above.

Lemma 4. *If C is a circuit of size t and degree d, there exists a weakly skew circuit computing the same polynomial and of size less than $t^{\log 2d}$.*

Proof. We will consider circuits with multiple output gates. The degree of such a circuit C is the maximal degree of a gate in C. If a circuit is weakly skew, for any multiplication gate one of the argument sub-circuit is independent from the rest of the circuit, in the sense that the values computed by its gates are not used elsewhere. A gate will be called *reusable* if it does not belong to the independent sub-circuit of a multiplication gate. In the case of Fig. 4, all gates are reusable except the leftmost input gate (x_1), the addition gate to which it is connected and the rightmost input gate (x_3).

Let us show by induction on n that for any integer d such that $2^n \leqslant d < 2^{n+1}$, for any (multiple output) circuit of size t and degree d, there exists a weakly skew circuit C' such that:

- the size of C' is at most $t^{\log 2d}$,
- for any gate α of C, there exists a reusable gate in C' which computes the polynomial computed by α in C.

If n is 0, the degree of C is 1 so that there are no multiplication gates in C. Thus C is weakly skew circuit and the property is true.

Suppose now that the property is true for all k strictly less than n, with $n \geqslant 1$. Consider C a circuit of size t and degree d, with $2^n \leqslant d < 2^{n+1}$. Call C_0 the circuit obtained by removing all gates of degree strictly greater than $\lfloor d/2 \rfloor$. Let t_0 be the size of C_0 and t_1 the number of gates of C of degree strictly greater than $\lfloor d/2 \rfloor$. We apply the induction hypothesis to C_0. This yields a circuit C_0' of size at most $t_0^{\log(2\lfloor d/2 \rfloor)} \leqslant t_0^{\log d}$. For any gate of C_0 there exists a reusable gate in C_0' computing the same polynomial. Consider now a multiplication gate of C of degree strictly greater than $\lfloor d/2 \rfloor$:

- if both its arguments are of degree at most $\lfloor d/2 \rfloor$, we add to C' a multiplication gate receiving arrows from a reusable gate of the first copy of C_0' and from a reusable gate of a new copy of C_0' (cf. Fig. 5),
- otherwise, since at least one of the arguments is of degree at most $\lfloor d/2 \rfloor$, the other having already been computed by a gate of C', we add to C' a multiplication gate receiving arrows from the gate of degree greater than $\lfloor d/2 \rfloor$ and from a reusable gate of a new copy of C_0'.

Addition gates are easy to deal with, we just connect them to reusable gates computing their arguments. The resulting circuit is weakly skew and satisfies the required conditions. Since $t = t_0 + t_1$, one can bound the size of C' as follows:

$$(t_1 + 1) \cdot t_0^{\log d} + t_1 \leqslant t \cdot t^{\log d} \leqslant t^{\log 2d}. \qquad \square$$

The classical proof of the completeness of the Determinant is to show a so-called universality property for formulas, namely that the polynomial computed by a formula of size s is a projection of the Determinant or the Permanent of a matrix of size polynomial in s. This is shown by building weighted graphs with adequate properties. Let G be an edge-weighted directed graph with two

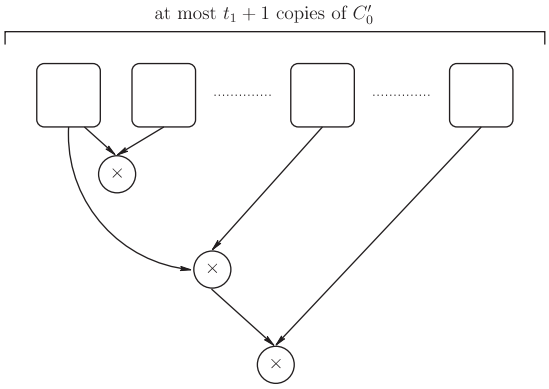

Fig. 5.

vertices s and t, the weight of a path from s to t is the product of the weights of the edges appearing in the path. The weight of (s, t) in G is the sum of the weights of all paths from s to t. To prove the universality lemmas one starts by building a graph whose weight is the polynomial computed by a formula (cf. [7] for example). We show here that the same construction can be done for weakly skew circuits.

Lemma 5. *Let C be a weakly skew circuit of size m, there exists an acyclic directed graph G, with two distinguished vertices s and t, such that: G is of size $m + 1$ and the weight of (s, t) in G is the polynomial computed by C.*

Proof. We will show a stronger result in the case of circuits with multiple outputs. We also keep the notion of reusable gates for a weakly skew circuit.

Let us show by induction on circuit size m that for any multiple output weakly skew circuit C there exists an acyclic directed graph G with a distinguished vertex s, satisfying the following conditions:

- G is of size at most $m + 1$,
- for any reusable gate α of C there exists a vertex t_α in G such that the weight of (s, t_α) in G is the polynomial computed by α in C.

A circuit of size $m = 1$ is made of one gate α with a (constant or variable) label u. The graph G with two vertices s and t_α and an edge (s, t_α) of weight u meets our requirements.

Suppose the above property is true for all integers strictly less than m ($m \geqslant 2$). Let C be a weakly skew circuit of size m and α one of its output gates.

If α is an input gate labeled u we just need to apply the induction hypothesis to the circuit C' with α removed. This yields a graph G' to which we add a new vertex t_α with an edge from s to t_α of weight u. Clearly the graph G thus obtained satisfies the necessary conditions.

If α is an addition gate, let C' be the circuit C without gate α. By induction hypothesis there exists a graph G' of size at most m. If α receives both its incoming arrows from one (necessarily reusable) gate β, there exists a vertex t_β in G' such that the weight of (s, t_β) in G' is the polynomial computed by β in C. We add a new vertex t_α and the edge (t_β, t_α) with weight 2 (cf. Fig. 6(a)). If α receives an arrow from two distinct gates β and γ, both necessarily reusable, there exist vertices

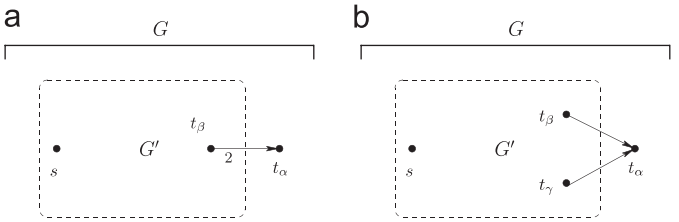

Fig. 6. Addition gate.

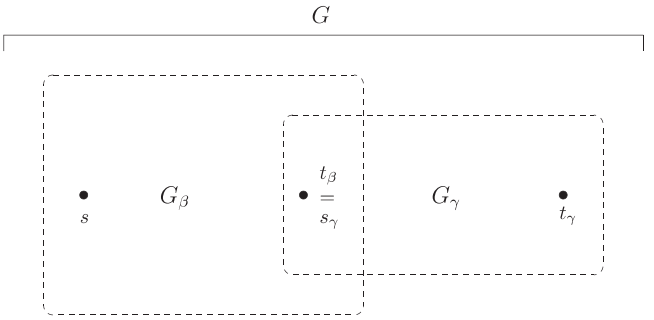

Fig. 7. Multiplication gate.

t_β and t_γ in G' such that the weights of (s, t_β) and (s, t_γ) in G' are the polynomials computed in C by β and γ, respectively. We then add a new vertex t_α to G' and the edges (t_β, t_α) and (t_γ, t_α) with weight 1 (cf. Fig. 6(b)). In both cases, the resulting graph G is of size at most $m + 1$ and satisfies the conditions.

If α is a multiplication gate, consider the distinct gates β and γ from which α receives an arrow. Suppose the sub-circuit C_γ is independent from the rest of the circuit, then the circuit C' obtained by removing α is composed of two disjoint circuits C_β and C_γ, of size m_β and m_γ such that $m = m_\beta + m_\gamma + 1$. Applying the induction hypothesis separately to C_β and to C_γ yields two graphs G_β and G_γ. In the first there are two vertices s and t_β such that the weight of (s, t_β) in G_β is the polynomial computed by β in C. In the second there are two vertices (s_γ, t) such that the weight of (s_γ, t) in G_γ is the polynomial computed by γ in C. We obtain G by identifying the vertices t_β and s_γ (cf. Fig. 7). G is of size at most $m_\beta + 1 + m_\gamma + 1 - 1 = m + 1$. The weight of (s, t) in G is clearly the product of the weight of (s, t_β) in G_β and of the weight of (s_γ, t) in G_γ, i.e. the polynomial computed by α in C. We have changed the value of the weights (s, v) for all vertices v in G_γ, but since these vertices were associated to the circuit C_γ whose gates are *not* reusable, the necessary properties still hold. □

From this construction we can show the universality of the Determinant for weakly skew circuits.

Lemma 6. *If f is a polynomial computable by a weakly skew circuit of size m, f is a projection of* DET_{m+1}.

Proof. From the graph G built in Lemma 5 we build a graph G' by identifying the vertices s and t and adding a loop to each vertex except $s = t$. Now consider the graph G'' obtained from G' by changing the weight of every edge which is not a loop into its opposite: if $A = (a_{i,j})$ is the matrix representing G', then the matrix representing G'' is the matrix B defined by $b_{i,j} = -a_{i,j}$ if $i \neq j$ and $b_{i,i} = a_{i,i}$ for all i. It can be shown that the Determinant of B is the polynomial $-f$. One need just add a last row and last column full of 0 except for the value in the bottom right hand corner which is -1. The Determinant of the resulting matrix is f. □

From this lemma and the fact that the Determinant is in VP and therefore in VQP one can show the completeness of the Determinant.

Theorem 3. *The Determinant is* VQP*-complete over any field.*

The following algebraic characterization of whether VNP is included in VQP is noted in [32] (it is shown in [7] that VQP is not included in VNP), it is an obvious consequence of Theorem 3.

Theorem 4. VNP $\subset$ VQP *iff the Permanent is a qp-projection of the Determinant.*

Now consider the families of polynomials (F_n), (G_n) and (H_n) defined by $F_n = \mathrm{Tr}(X^n)$, $G_n = \mathrm{Tr}(X_1 \cdots X_n)$ and $H_n = \mathrm{Tr}(\mathrm{DET}(X) \cdot X^{-1})$, where Tr is the trace, and X or X_i are matrices with n^2 variables. The same technique can be used to show their completeness for VQP, thus providing a full answer to Conjecture 8.1 from [7]. One just needs to show a universality result and computability by weakly skew circuits of quasi-polynomial size. We will only describe the steps for (F_n), since it is the case missing from [6], which gives a partial answer to the conjecture. It is easy to show from the inductive definition that matrix powering can be computed by weakly skew circuits, in fact one can show a stronger result using skew circuits (cf. [25]). To show universality we use the generic construction of Lemma 5 and then modify the resulting graph. The construction is much more involved than for the Determinant. It could be simplified if we just wanted to prove the completeness of computing the (1, 1)-coefficient of the power of a matrix, and this would be our choice because the universality of matrix powering will be used later to show the equivalence of skew and weakly circuits. As it is we show the more difficult result with the trace to answer the conjecture.

Lemma 7. *If f is a polynomial computable by a weakly skew circuit of size m, f is a projection of F_{2m+3} or F_{2m+5}.*

Proof. Let f be a polynomial computed by a weakly skew circuit C. Define a *walk* of length k in a directed graph as a sequence of vertices $(t_1, \ldots, t_k)$ such that the edges (t_i, t_{i+1}) and the edge (t_k, t_1) belong to the graph. A walk may go through a given vertex several times. The vertex t_1 is called the *origin* of the walk. The weight of a walk is the product of the weights of its edges. The k-weight of a graph G is the sum of the weights of all walks of length k.

Let X be the matrix with entries $x_{i,j}$ $(1 \leqslant i, j \leqslant n)$, it is easy to show that the polynomial $\mathrm{Tr}(X^n)$ is equal to

$$\sum_{1 \leqslant k_1, \ldots, k_n \leqslant n} x_{k_1,k_2} \ldots x_{k_{n-1},k_n} x_{k_n,k_1}.$$

If we interpret the matrix X as the adjacency matrix of a graph G, we can see that $\mathrm{Tr}(X^n)$ is the n-weight of G. We therefore wish to build a graph of size l whose l-weight is the polynomial f.

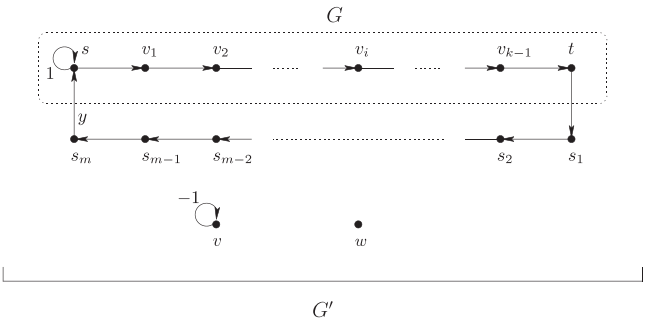

Fig. 8.

Lemma 5 yields an acyclic directed graph G of size $m+1$, with vertices s and t such that the weight of (s,t) is the polynomial f. We start by adding m vertices $s_1,\dots,s_m$, and the edges (t,s_1), (s_i,s_{i+1}), all with weight 1, and finally the edge (s_m,s), whose weight is a new variable y. We also add a loop of weight 1 to the vertex s, a vertex v with a loop of weight -1 and an isolated vertex (cf. Fig. 8). The size of the resulting graph G' is $2m+3$. Let us study the walks of G of length $2m+3$.

There is a unique walk of length $2m+3$ which consists in looping on the vertex v. Because $2m+3$ is an odd integer, its weight is -1.

There is a unique walk of length $2m+3$ which consists in looping on the vertex s. Its weight is 1.

Let $\tau = s, v_1, \dots, v_{k-1}, t$ be a path from s to t of length k in G (and therefore in G'). The vertex v_i is the origin of a unique walk of length $2m+3$ going through τ. It consists in going to t via τ (length $k-i$), then going to s via the vertices s_i (length $m+1$), looping $m+2-k$ times in s (one can check that $m+2-k \geqslant 0$) finally returning to v_i via τ (length i). The total length is $2m+3$. The path τ yields a unique walk for each of the vertices $t, s_1, \dots, s_m$. For each of the vertices, any other walk would include going around twice, thus its length would be at least $2(m+2)$, which is strictly greater than $2m+3$. There are also $m+3-k$ walks with s as origin, of length $2m+3$ and going through τ, depending on whether one loops $0, 1, \dots$ or $m+2-k$ times in s before going on τ. All these walks have the same weight, namely the weight of τ multiplied by y. There are thus $2m+3$ walks of length $2m+3$ associated with τ. The $(2m+3)$-weight of G' is therefore:

$$(-1) + 1 + \sum_{\substack{\tau \text{ path} \\ \text{from } s \text{ to } t}} y(2m+3) \cdot \text{weight}(\tau).$$

In characteristic 0, we just have to substitute $(2m+3)^{-1}$ for the variable y to get the polynomial f.

In characteristic $p > 0$, we need to be a little more careful. For a fixed m the same construction can be done if p does not divide $2m+3$, and then use the inverse of $2m+3$ as above. If p divides $2m+3$, then p is strictly greater than 2, and p does not divide $2m+5$. We follow the above construction but add two vertices s_{m+1} and s_{m+2}. □

Theorem 5. *The families* (F_n), (G_n) *and* (H_n) *are* VQP*-complete over any field.*

5.2. *The class* $\mathrm{VP}_{\mathrm{ws}}$

We have already said that [25] gives an excellent account of the complexity of the Determinant, which can be immediately transposed into Valiant's setting. In fact, Toda defines the very natural class DET(poly) of polynomial families which can be expressed as the Determinant of a sequence of matrices (with variable or constant entries) of polynomially bounded size. This class is shown to be characterized by skew arithmetic circuits, and equivalently by weakly skew arithmetic circuits. Let us rename this class $\mathrm{VP}_{\mathrm{ws}}$ in Valiant's framework and define it directly by weakly skew circuits.

Definition 11. A sequence of polynomials (f_n) belongs to the class $\mathrm{VP}_{\mathrm{ws}}$ if it is represented by a sequence of weakly skew circuits of polynomially bounded size.

This class is a much more natural candidate to capture the complexity of the Determinant than VQP. The universality Lemma 6 combined with the following proposition on the computation of the Determinant by weakly skew circuits proves the completeness of the Determinant. Note that this completeness is under standard p-projections, which is one reason we suggest this class be preferred to VQP.

Proposition 5. (DET_n) *can be computed by a sequence of weakly skew circuits of polynomial size.*

Proof. We recall here Berkowitz's algorithm [5], as described in [26]. We claim that this computation can be done by a weakly skew circuit of polynomial size.

Let B_k $(0 \leqslant k \leqslant n-1)$ be the principal $(n-k) \times (n-k)$ minor of X. Define the $(n-k) \times 1$ matrix C_k and the $1 \times (n-k)$ matrix D_k for $1 \leqslant k \leqslant n-1$ as follows:

$$B_{k-1} = \begin{pmatrix} B_k & C_k \\ D_k & X_{n-k+1,n-k+1} \end{pmatrix}.$$

For each k $(1 \leqslant k \leqslant n)$ define T_k as the following $(n+2-k) \times (n+1-k)$ matrix:

$$(T_k)_{i,j} = \begin{cases} 0 & \text{if } i > j+1, \\ -1 & \text{if } i = j+1, \\ X_{n-k+1,n-k+1} & \text{if } i = j, \\ D_k B_k^{j-i-1} C_k & \text{if } i < j. \end{cases}$$

Then the coefficients of the characteristic polynomial are given by the $(n+1) \times 1$ matrix $\prod_{k=1}^{n} T_k$, where the $(1, 1)$ coefficient is the Determinant.

Because a product of matrices can be computed by a weakly skew circuit of polynomial size, each T_k can be computed by an weakly skew circuit of size polynomial in n (each entry of a given T_k is computed separately). In computing the product of the T_k we will need at most $(n+1)$ distinct copies of each T_k, so the overall size of the weakly skew circuit stays polynomial in n. □

Theorem 6. *The sequence* (DET_n) *is* $\mathrm{VP}_{\mathrm{ws}}$*-complete over any field.*

Proof. This is a consequence of Lemma 6 (to show that any polynomial sequence in $\mathrm{VP_{ws}}$ is a p-projection of the Determinant) and of Proposition 5 (to show that the Determinant belongs to $\mathrm{VP_{ws}}$). □

Defining $\mathrm{VP_{ws}}$ via weakly skew circuits puts it naturally between $\mathrm{VP_e}$ and VP. The proof of completeness for the Determinant is easier because it is simpler to show that it can be computed by weakly skew circuits than skew circuits. Moreover, if we follow this order, we can use the completeness of matrix powering for $\mathrm{VP_{ws}}$ (this can be proved by following the same proof strategy as for VQP, using Lemma 7), to get an immediate proof of the characterization of $\mathrm{VP_{ws}}$ by skew arithmetic circuits, thus avoiding the more technical constructions in [25]. Indeed, any family in $\mathrm{VP_{ws}}$ is a p-projection of (F_n). As noticed before, (F_n) can be computed by sequences of *skew* circuits of polynomial size. The strict nature of p-projections thus yields polynomial size skew circuits for any family in $\mathrm{VP_{ws}}$, including the Determinant. One can also show the $\mathrm{VP_{ws}}$-completeness of the families (G_n) and (H_n).

Theorem 7. *The families* (F_n), (G_n) *and* (H_n) *are* $\mathrm{VP_{ws}}$*-complete over any field.*

Proposition 6. *A sequence of polynomials* (f_n) *belongs to the class* $\mathrm{VP_{ws}}$ *if it is represented by a sequence of skew circuits of polynomially bounded size.*

5.3. The Permanent and the Determinant

The completeness of the Determinant entails another complexity theoretic characterization of the relation between the Permanent and the Determinant, similar to 4, but more natural. Both these results are interesting in that they relate a question from computational complexity to an easily stated mathematical problem.

Theorem 8. *The Permanent is a p-projection of the Determinant iff* $\mathrm{VP_{ws}} = \mathrm{VNP}$.

Several articles have been written on the links between the Permanent and the Determinant, going back to Pólya [23], who asks whether one can change the sign of the entries of a $\{0, 1\}$ matrix so that the Determinant of the resulting matrix is the Permanent of the original one. A result such as the one above indicates that even the more general procedure of computing a Permanent as the Determinant of a polynomially bigger matrix is probably not always possible. There are, however, some links between Permanent and Determinant computations. For instance, it has been shown [12] that the Permanent of a graph of genus g can be computed as a linear combination of 4^g Determinants. For graphs of small genus this yields a feasible computation. In the case of circulant matrix with three non-zero entries per row, the Permanent can be expressed as a linear combination of just 4 Determinants [8]. Could the strategy of using a linear combination of a polynomial number of Determinants work in the general case?

To answer this question, we will use the notion of being *linearly closed*, as defined in [7].

Definition 12. A sequence (g_n) is called *linearly closed* if any linear combination $\sum_{k=1}^{n} \lambda_k g_{i_k}$ is a projection of some g_m, where m is polynomially bounded in the number n of terms and $\max_k i_k$. Hereby the sets of variables of the g_i are supposed to be (made) disjoint for distinct k.

Bürgisser then asks whether the Determinant family has this property (Problem 3.2). The answer is positive and the completeness of the Determinant for the class $\mathrm{VP_{ws}}$ provides a very simple proof.

Proposition 7. *The Determinant is linearly closed.*

Proof. Suppose that $\lambda_1, \ldots, \lambda_n$ are variables from a ring, and that $A_1, \ldots, A_n$ are matrices of size $s_1, \ldots, s_n$, respectively, whose entries are independent variables. We wish to express the sum $\lambda_1 \operatorname{DET}(A_1) + \cdots + \lambda_n \operatorname{DET}(A_n)$ as the Determinant of a matrix of small size.

We know that there exists a polynomial r such that the Determinant of a matrix of size s is computed by a weakly skew circuit of size $r(s)$. Thus one can build a weakly skew circuit of size $2n - 1 + \sum_{k=1}^{n} r(s_k)$ which computes the linear combination above. The polynomial computed by this circuit is by Lemma 6 a Determinant of a matrix of size $2n + \sum_{k=1}^{n} r(s_k)$. □

As a consequence, the strategy of expressing a Permanent as a linear combination of Determinants is also likely to fail in general.

Theorem 9. *The Permanent can be expressed as a linear combination of a polynomial number of Determinants of polynomial size iff* $\mathrm{VP}_{\mathrm{ws}} = \mathrm{VNP}$.

Mignon and Ressayre [20] study the "determinental complexity" of the Permanent, i.e. the smallest size of a matrix whose Determinant computes a given Permanent, and give a quadratic lower bound. Using Ryser's algorithm [24], one can give an exponential upper bound (via formulas). Combining the work of Galluccio and Loebl [12] and the fact that the Determinant is linearly closed, we can obtain an upper bound depending on the genus of the graphs considered.

To summarize, we have considered the increasing expressive power of the following sequence of models, when the size is polynomially bounded: formulas, weakly skew circuits, MD circuits. One of the reasons VQP was considered a "good" class is that if we allow a quasi-polynomially bounded size, all these classes are equal (cf. [32]). This collapse also occurs if we polynomially bound the depth rather than the size. And, as we shall see in the next section, in the uniform case the resulting class characterizes VNP.

6. Characterizing uniform VNP

We wish to compare the respective expressive power of Boolean sums in front of a circuit of polynomial size and degree (VNP) on the one hand and of circuits of polynomial depth and degree on the other. This is related to the characterization of #P via circuits of polynomial depth and degree in [29]. We will show that a similar theorem holds for a uniform version of Valiant's algebraic classes.

At the non-uniform level it is easy to see that circuits of polynomial depth and degree are at least as powerful as VNP. Indeed a sequence in VNP is defined from a sequence in VP which is represented by circuits of polynomial size and degree, and therefore polynomial depth and degree. By computing in parallel all the values of these circuits for all Boolean strings of appropriate length and then summing, we get a circuit of polynomial depth and degree. The summation can be done in polynomial depth because there is a simply exponential number of gates to sum.

For the converse we would like to express the polynomial computed by a circuit of polynomial depth and degree as a sum of the values of a circuit of polynomial size and degree. We will use the same strategy as when proving the equality of VNP and $\mathrm{VNP}_{\mathrm{e}}$. The value of a circuit is written as the sum of the values of its parse trees. Although there is no polynomial bound on the size of our original circuit, the constraints on depth and degree give us a constraint on the size of the parse trees, as noticed in [29].

Lemma 8. *If C is a circuit of depth p and degree d, any parse tree of C is of size less than pd.*

Proof. By induction on depth. □

However, we also need to recognize efficiently whether a Boolean string encodes a parse tree or not (previously we used the sub-circuit property because our circuits were of polynomial size). This will be made possible by the second ingredient, uniformity. We will use the condition given in [29]. Define the *direct connection language* of a sequence of circuits C_n as the set of strings of the form $\langle n, g, y, p\rangle$ such that either (i) g is an addition gate in C_n and y is an input of g, or (ii) g is a multiplication gate in C_n and y is a left or right input of g depending on p, or (iii) g is a gate name in C_n and y is the type of g. A sequence of circuits C_n is DLOGTIME-uniform if its direct connection language can be recognized by a deterministic Turing machine in time logarithmic in the size of the circuits. In our case, with circuits of exponential size, it means that we can get information on an arrow or a gate in polynomial time.

Let us now define the uniform classes we have mentioned. For Valiant's classes, P-uniformity is the most natural notion, meaning that the circuit C_n is produced by a Turing machine in polynomial time upon input of n in unary. We may either consider constant-free circuits (for instance using just the constant $0, 1, -1$), or allow sequences of circuits $(C_n(\bar{x}, a_1, \ldots, a_k))$ which use a fixed set of constants $A_1, \ldots, A_k$ (the set of constants depends on the sequence). In any case, this means we will work with the constant-free circuits, which can be encoded by Boolean words.

Definition 13. A sequence of polynomials is in the class $\mathrm{VP_u}$ if it is represented by a P-uniform sequence of circuits of polynomial size and degree.

A sequence of polynomials (f_n) belongs to $\mathrm{VNP_u}$ if there exists a polynomial p and a sequence $g_n \in \mathrm{VP_u}$ such that $f_n(\bar{x}) = \sum_{\bar{\varepsilon}\in\{0,1\}^{p(|\bar{x}|)}} g_n(\bar{x}, \bar{\varepsilon})$.

Theorem 10. *A sequence of polynomials (f_n) belongs to the class* $\mathrm{VNP_u}$ *iff it can be represented by a* DLOGTIME*-uniform sequence of circuits of polynomial depth, degree and number of input variables.*

Proof. (1) Any sequence in $\mathrm{VNP_u}$ can be represented by DLOGTIME-uniform sequence of circuits of polynomial depth, degree and number of input variables.

Let (f_n) be a sequence in $\mathrm{VNP_u}$. Then there exists a uniform sequence of circuits (G_n) of polynomial size and degree such that $f_n(\bar{x}) = \sum_{\bar{\varepsilon}} g_n(\bar{x}, \bar{\varepsilon})$. Let $p(n)$ be the length of the Boolean words $\bar{\varepsilon}$. We will build a sequence of circuits (C_n). We start by placing in parallel $2^{p(n)}$ copies of the circuit G_n. Each copy is indexed by a Boolean word $\bar{\eta}$ of length $p(n)$. We replace the $\bar{\varepsilon}$ inputs of the copy indexed by $\bar{\eta}$ by the values of $\bar{\eta}$. We then add an addition tree of depth $p(n)$ which sums all the outputs of the copies of G_n. The size of the resulting circuit is simply exponential, while its depth and its degree are polynomial. Each gate in a copy of G_n is encoded by giving both its encoding in G_n and the index $\bar{\eta}$ of the copy. Each gate in the addition tree is encoded by the depth of its row and position in the row.

The following machine can recognize the direct connection language of the resulting sequence of circuits: it uses n to build the circuit G_n (in polynomial time); it can easily answer queries about gates in the addition tree; to answer queries about gates in copies of G_n it uses the index $\bar{\eta}$ of the copy (copies) concerned and the encoding of G_n computed at the onset to answer. All this can be done in polynomial time, and thus in time logarithmic with regard to the size of the circuit sequence.

(2) Any DLOGTIME-uniform sequence of circuits of polynomial depth, degree and number of input variables belongs to $\mathrm{VNP}_{\mathrm{u}}$.

From circuits to formulas: It is easy to transform any circuit of polynomial depth and degree into a formula of polynomial depth and degree by duplicating gates. However, we must make sure that the direct connection language of the formula is still checkable in time logarithmic in the size of the circuit. We will assume that the resulting circuit is of exponential size, and thus our machine should work in polynomial time. We also assume that the encoding of the gates of the original circuit lets us recognize the output gate of the circuit and the position (left or right) of the arguments (this is a consequence of the *admissible encodings* defined in [31]; while not being as strict, we will keep these properties of the encoding). The gates in our formulas will be named as they were in the original circuit, but prefixed with the path coming from the output (the list of gates names), with flags L or R to indicate which edge was chosen. Because the depth is polynomial, and gate names are of polynomial length, the resulting encoding is still polynomial. When testing the direct connection language of the formula, the machine starts by checking that the path does exist and that it leads to the gate being considered, by querying the original machine. One can then check the type of the gate or its connection with other gates using the original machine. (This argument basically shows that uniform sequences of formulas are equivalent to circuits for polynomial depth and degree.)

Checking parse trees: Consider now our DLOGTIME-uniform sequence (C_n) of formulas of polynomial depth and degree. We will encode a parse tree of a formula as a list of the gates in the parse tree, with the condition that the gates be listed in lexicographic order. As stated in Lemma 8, these lists are still of polynomial size. There is a Turing machine which given n and a Boolean word $\bar{t}$ checks in polynomial time whether $\bar{t}$ encodes a parse tree of C_n: it does so using the conditions given in Section 4.1 (and also checking the ordering of the gates):

(1) Check that the output gate belongs to the parse tree.
(2) For a multiplication gate, check that both arguments also appear (by testing the argument relation of the direct connection language with all the other gates in the list).
(3) For an addition gate, check that exactly one argument appears in the list.
(4) Check that each gate (except the output) is an argument to at least one other gate in the list.

The machine then outputs the list of the input gates of the parse tree. All these steps can be done in time polynomial in n because checking the direct connection language can be done in polynomial time, and the parse tree is of polynomial size.

Building Boolean circuits: This Turing machine can then be unfolded into a (P-uniform) sequence of (multiple-output) Boolean circuits $D_n(\bar{t})$, whose output is the list of input gates of $\bar{t}$, if $\bar{t}$ encodes a parse tree of C_n, and the Boolean word $\bar{0}$ otherwise. This Boolean circuit can be simulated by an arithmetic one, but there is no guarantee that the degree will be polynomial. Before converting to an arithmetic circuit, we therefore make the following transformation. For each gate α in the circuit we add a new variable y_α. We then proceed in the manner of [22, p. 151]: if α is an input gate, then y_α is equal to the input variable of α; if α is the conjunction (resp., disjunction) of gates β and γ in the circuit D_n, we replace it by the formula $y_\alpha = y_\beta \wedge y_\gamma$ (resp., $y_\alpha = y_\beta \vee y_\gamma$). Let Φ be the resulting set of formulas (in variables $\bar{y}$ and $\bar{t}$). Given some Boolean inputs to the variables $\bar{t}$, there is only one possible value for all the y variables, which is the value of the associated gates in D_n. We then build a new Boolean circuit. First we compute in parallel the value of the formulas in Φ. Then we compute their conjunction by a tree of conjunctions of logarithmic depth (the cardinality of Φ is polynomial). Let ρ be the gate at the end of the conjunction tree. Finally, for every output gate α in D_n, we will have a new output gate

which is the conjunction of ρ and y_α. Thus we have a Boolean circuit $D'_n(\bar{t}, \bar{y})$ of logarithmic depth such that for any Boolean input $\bar{t}$ which encodes a parse tree, there is exactly one Boolean word $\bar{b}$ for which the output encodes the list of input gates of the parse tree, otherwise the output is the Boolean word $\bar{0}$.

Building arithmetic circuits: We then simulate this Boolean circuit by an arithmetic circuit, whose degree is polynomial because its depth is logarithmic. We add gates which compute the product of the variables from the list of input gates (it is easy to show by induction that there exists a P-uniform sequence of circuits of polynomial size and degree which takes the variables $x_0, \ldots, x_{2^n-1}$ and $\bar{n}$ and $\bar{s}$ as input, and outputs the variable x_i such that i is the integer encoded by $\bar{s}$). We now have a circuit $E_n(\bar{x}, \bar{t}, \bar{y})$ such that the polynomial computed by C_n is the sum over Boolean words $\bar{t}, \bar{y}$ of $E_n(\bar{x}, \bar{t}, \bar{y})$. From the above construction one can see that the circuit E_n can be built by a Turing machine in time polynomial in n. □

Note the similarity of this characterization with the characterization of #P by Venkateswaran [29]. In both cases the class characterized is uniform. In our description of the proof strategy we emphasize the role played by uniformity. What happens in the non-uniform case?

We will use a converse of Valiant's criterion (cf. [7]) to answer this question. Valiant gave a criterion for showing that specific sequences of polynomials belong to VNP, the rough idea being that sequences whose coefficient function is in $\sharp\mathrm{P}/poly$ belong to VNP. One can show a converse of this theorem by using the coefficient function (we will not give details here, this is also noticed in [21, p. 14] and can be proved using techniques in [18]). Such a converse states that if we have a family of functions (f_n), where $f_n : \{0,1\}^n \to \left[0, \ldots, 2^{p(n)}\right]$, and if we define $g_n(x_1, \ldots, x_n) = \sum_{\bar{\varepsilon}\in\{0,1\}^n} f_n(\bar{\varepsilon})\, x_1^{\varepsilon_1} \ldots m x_n^{\varepsilon_n}$, then $(g_n) \in \mathrm{VNP}$ implies $(f_n) \in \sharp\mathrm{P}/poly$.

Now consider any sequence of functions (f_n), with $f_n : \{0,1\}^n \to [0, \ldots, 2^{p(n)}]$ and view it as the coefficient function of a polynomial sequence $g_n(\bar{x}) = \sum_{\bar{\varepsilon}} f_n(\bar{\varepsilon}) x_1^{\varepsilon_1} \ldots m x_n^{\varepsilon_n}$. We can compute in polynomial depth these monomials and the integer coefficients and just sum them, so that the sequence (g_n) can be computed by a sequence of circuits of polynomial depth and degree. If the hypothesis is true in this non-uniform case, then (g_n) belongs to VNP. Thus (f_n) belongs to $\sharp\mathrm{P}/poly$. However, there exists functions which are not in $\mathrm{PSPACE}/poly$, and thus not in $\sharp\mathrm{P}/poly$, which contradicts the assumption. Thus the uniformity condition in this proof is not insignificant, but rather an essential ingredient of the proof.

7. Conclusion

We have shown in this paper that different classes in Valiant's framework can be defined via a hierarchy of circuits of polynomial size, from formulas to weakly skew circuits to MD circuits, and that all these restrictions become equivalent for polynomial depth and (in the uniform case) define the class VNP. These characterizations came with new results and new proofs of old results. In our view, one important aim of this paper is to bring attention to the work of Toda [25] and suggest the adoption of the class $\mathrm{VP}_{\mathrm{ws}}$.

To stress the importance of this class we would like to find other complete polynomials. For any polynomial family which is shown to be in VP we should check if one can show that it is $\mathrm{VP}_{\mathrm{ws}}$-complete. For instance the generating function of trees (cf. [7, Chapter 3]) is reduced to the Determinant and thus belongs to $\mathrm{VP}_{\mathrm{ws}}$. It would be interesting to know whether it is complete. The class $\mathrm{VP}_{\mathrm{ws}}$ can also be seen as capturing the computational power of directed acyclic graphs with weights. When one builds a graph as in Lemma 5 starting from a formula, the resulting directed

graph is *series–parallel*, a property which has been studied in the context of task ordering or parametrized complexity. The question of the respective power of weakly skew circuits versus formulas, which is the question of whether the Determinant can be computed by weakly skew circuits, is exactly the problem of whether a general st-dag can be transformed into a series–parallel one of same weight without an explosion of its size. We think it would be interesting to study these links.

As for the characterization of VP, it could help us find a natural complete problem. A good strategy for this is to look more closely at the class LOGCFL, its properties and complete problems. One possibility would be to use the tensor formulas defined in [10], where an example of a LOGCFL-complete problem is given. Indeed, the different kinds of tensor formulas seem to fit very well with the classes $\mathrm{VP}_{\mathrm{ws}}$, VP and VNP, and we hope to explore this link in a future work.

One last obvious question is the separation of VP and $\mathrm{VP}_{\mathrm{ws}}$, or in other words whether an MD circuit can be transformed into a weakly skew circuit without an exponential blow-up in size. If the answer is positive, then the classes VP and $\mathrm{VP}_{\mathrm{ws}}$ are equal, the Determinant is VP-complete, and interestingly the theorem stating that $\mathrm{VNP} = \mathrm{VNP}_{\mathrm{e}}$ is not necessary to prove the completeness of the Permanent. On the other hand, if sequences in VP do not admit sequences of weakly skew circuits of polynomial size, then the classes VP_{e} and VP are distinct, an answer to a major open question, and the Determinant is not VP-complete. Thus the restrictions imposed on multiplications seem to be a crucial point in Valiant's complexity classes, although answers to the above questions will be hard to come by.

References

[1] E. Allender, Arithmetic circuits and counting complexity classes, in: J. Krajicek (Ed.), Complexity of Computations and Proofs, Quaderni di Matematica, vol. 13, Seconda Universita di Napoli, 2004, pp. 33–72.

[2] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, P. B. Miltersen, On the complexity of numerical analysis, in: Proceedings of the 21st Annual IEEE Conference on Computational Complexity (July 16–20, 2006). CCC. IEEE Computer Society, Washington, DC, pp. 331–339.

[3] E. Allender, J. Jiao, M. Mahajan, V. Vinay, Non-commutative arithmetic circuits: depth reduction and size lower bounds, Theoret. Comput. Sci. 209 (1–2) (1998) 47–86.

[4] L. Babai, L. Fortnow, Arithmetization: a new method in structural complexity theory, Comput. Complexity 1 (1) (1991) 41–66.

[5] S.J. Berkowitz, On computing the determinant in small parallel time using a small number of processors, Inf. Process. Lett. 18 (3) (1984) 147–150.

[6] M. Bläser, Complete problems for Valiant's class of qp-computable families of polynomials, in: COCOON '01: Proceedings of the 7th Annual International Conference on Computing and Combinatorics, Springer, London, UK, 2001, pp. 1–10.

[7] P. Bürgisser, Completeness and reduction in algebraic complexity theory, Algorithms and Computation in Mathematics, vol. 7, Springer, Berlin, 2000.

[8] B. Codenotti, G. Resta, Computation of sparse circulant permanents via determinants, Linear Algebra Appl. 355 (2002) 15–34.

[9] C. Damm, $\mathrm{DET} = L^{\#L}$, Informatik-Preprint 8, Humboldt-Universität zu Berlin, 1991.

[10] C. Damm, M. Holzer, P. McKenzie, The Complexity of Tensor Calculus, Comput. Complexity 11 (1–2) (2002) 54–89.

[11] R. Diestel, Graph Theory, 3rd ed., Graduate Texts in Mathematics, vol. 173, Springer, Berlin, 2005.

[12] A. Galluccio, M. Loebl, On the theory of Pfaffian orientations. I. Perfect matchings and permanents, Electron. J. Combin. 6 (1999) 1–18.

[13] N. Immerman, S. Landau, The complexity of iterated multiplication, Inf. Comput. 116 (1) (1995) 103–116.

[14] M. Jerrum, M. Snir, Some exact complexity results for straight-line computations over semirings, J. ACM 29 (3) (1982) 874–897.

[15] V. Kabanets, R. Impagliazzo, Derandomizing polynomial identity tests means proving circuit lower bounds, in: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, San Diego, CA, USA, June 9–11, 2003, pp. 355–364.

[16] E. Kaltofen, Uniform closure properties of P-computable functions, in: STOC '86: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, ACM Press, New York, NY, USA, 1986, pp. 330–337.
[17] P. Koiran, Valiant's model and the cost of computing integers, Comput. Complexity 13 (3–4) (2005) 131–146.
[18] G. Malod, Polynômes et coefficients, Ph.D. Thesis, Université Claude Bernard Lyon 1, 2003, ⟨URL: http://tel.ccsd.cnrs.fr/tel-00087399⟩.
[19] P. McKenzie, H. Vollmer, K.W. Wagner, Arithmetic circuits and polynomial replacement systems, in: S. Kapoor, S. Prasad (Eds.), FSTTCS, of Lecture Notes in Computer Science, vol. 1974, Springer, Berlin, 2000, pp. 164–175.
[20] T. Mignon, N. Ressayre, A quadratic bound for the determinant and permanent problem, Internet. Math. Res. Notes. (79) (2004) 4241–4253.
[21] S. Périfel, Polynômes donnés par des circuits algébriques et généralisation du modèle de Valiant, Master's Thesis, École Normal Supérieure de Lyon, France, June 2004, ⟨URL:www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2004/DEA2004-07.ps.gz⟩.
[22] B. Poizat, Les Petits Caillloux Nur Al-Mantiq Wal-Ma'rifah, vol. 3, Aléas, Lyon, 1995.
[23] G. Pólya, Aufgabe 424, Arch. Math. Phys. 20 (1913) 271.
[24] J.R. Ryser, Combinatorial mathematics, The Carus Mathematical Monographs, No. 14, The Mathematical Association of America, 1963.
[25] S. Toda, Classes of arithmetic circuits capturing the complexity of computing the determinant, IEICE Trans. Inf. Syst. E75-D (1992) 116–124.
[26] L.G. Valiant, Why is Boolean complexity theory difficult? in: Boolean Function Complexity (Durham, 1990), London Mathematical Society Lecture Note Series, vol. 169, Cambridge University Press, Cambridge, 1992, pp. 84–94.
[27] L.G. Valiant, S. Skyum, S. Berkowitz, C. Rackoff, Fast parallel computation of polynomials using few processors, SIAM J. Comput. 12 (4) (1983) 641–644.
[28] H. Venkateswaran, Properties that characterize LOGCFL, J. Comput. Syst. Sci. 43 (2) (1991) 380–404.
[29] H. Venkateswaran, Circuit definitions of nondeterministic complexity classes, SIAM J. Comput. 21 (4) (1992) 655–670.
[30] H. Venkateswaran, M. Tompa, A new pebble game that characterizes parallel complexity classes, SIAM J. Comput. 18 (3) (1989) 533–549.
[31] H. Vollmer, Introduction to Circuit Complexity: A Uniform Approach, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 1999.
[32] J. von zur Gathen, Feasible arithmetic computations: Valiant's hypothesis, J. Symb. Comput. 4 (2) (1987) 137–172.