A la recherche de la definition de la complexite d'espace pour le calcul des polynomes a la maniere de Valiant

Bruno Poizat

The Journal of Symbolic Logic / Volume 73 / Issue 04 / December 2008, pp 1179 - 1201
DOI: 10.2178/jsl/1230396913, Published online: 12 March 2014

Link to this article: http://journals.cambridge.org/abstract_S0022481200003947

How to cite this article:
Bruno Poizat (2008). A la recherche de la definition de la complexite d'espace pour le calcul des polynomes a la maniere de Valiant . The Journal of Symbolic Logic, 73, pp 1179-1201 doi:10.2178/jsl/1230396913

THE JOURNAL OF SYMBOLIC LOGIC
Volume 73, Number 4, Dec. 2008

A LA RECHERCHE DE LA DEFINITION DE LA COMPLEXITE D'ESPACE POUR LE CALCUL DES POLYNOMES A LA MANIERE DE VALIANT

BRUNO POIZAT

Résumé. Nous définissons une classe de suites de polynômes, calculés par des circuits de complexité polynomiale comprenant des additions, des soustractions, des multiplications et des sommations de Valiant. Nous montrons que cette classe est close pour la prise de la fonction-coefficient, définie au paragraphe 3 de cet article ; nous en déduisons l'existence d'un circuit de complexité $72.n^2$, calculant le coefficient binomial de deux nombres de n chiffres, donnés en base 2. Il est par ailleurs facile de construire un circuit de complexité $17.n + 2$ calculant la factorielle d'un nombre de n chiffres. La présence de $2.n$ sommations d'effet exponentiel dans chacun de ces circuits en affecte gravement l'intérêt pratique. Il est peu probable, ou du moins peu souhaitable, qu'on puisse éliminer ces sommations sans explosion, car cela provoquerait la catastrophe cryptographique que redoutent tous les banquiers ; néanmoins, nous ne savons pas séparer la classe définie ici de celle des suites de polynômes calculables en un nombre polynomial d'opérations arithmétiques. Cela n'a rien de surprenant, vu la très grande affinité qu'elle a avec la classe PSPACE : nous montrons en effet que cette classe est identique à la classe VPSPACE, définie antérieurement par Koiran et Perifel, qui apparaît ici sous une forme bien plus maniable que l'originale.

§1. Classes de complexité, façon Valiant. Dans cette section liminaire, nous rappelons la définition des objets utilisés pour l'étude de la complexité de calcul des polynômes à la manière de Valiant, et nous précisons les conventions que nous adoptons. Pour plus de détails sur le sujet, on pourra consulter [Bürgisser, 2000].

Un circuit arithmétique C est un graphe orienté fini, sans cycles orientés, à une seule sortie, dont les entrées sont étiquetées par des variables ou par des constantes, et dont les autres sommets, appelés portes, reçoivent exactement deux flèches et sont étiquetés par l'addition, ou bien par la multiplication. Le sous-circuit Cp associé à la porte p est formé des portes de C situées sur un chemin arrivant à p.

Sauf de manière épisodique, nous n'employons que la constante -1.

Contrairement à ce que conviennent certains, nos circuits arithmétiques ne font pas de divisions ; ils décomposent les soustractions en une multiplication par la constante -1 suivie d'une addition.

Nous distinguons deux sortes de variables ; les premières ne prennent que les valeurs 0 ou 1 ; elles sont qualifiées de booléennes, bien que leur valeur soit un entier, et non pas un entier modulo 2 ; les autres variables sont libres de prendre les valeurs de leur choix.

Received January 5, 2007.

Mots-clefs. Polynômes, circuits arithmétiques, complexité, opérations, quantifications, P et PSPACE, Valiant.

0022-4812/08/7304-0006/$3.30

S'il n'y a pas de variables booléennes, le circuit calcule un polynôme $f(\underline{x})$ à coefficients entiers relatifs. S'il y en a, il calcule ce que nous appelons une fonction-polynôme, qui au uplet booléen $\underline{u}$ associe un polynôme $f_{\underline{u}}(\underline{x})$, que nous pouvons tout aussi bien noter $f(\underline{u}, \underline{x})$. Si le circuit ne comporte pas de variables non-booléennes, ce polynôme est une constante, et nous parlerons alors de fonction numérique. Une fonction numérique est dite booléenne si elle ne prend que les valeurs 0 ou 1.

Toute fonction-polynôme a pour origine un polynôme dont on a spécialisé booléennement certaines variables. Cela se montre facilement par récurrence sur la longueur de $\underline{u}$, en itérant la formule $f(\underline{u}, v, \underline{x}) = v.f(\underline{u}, 1, \underline{x}) + (1 - v).f(\underline{u}, 0, \underline{x})$. Ce polynôme n'est bien sûr pas unique, puisque seules comptent les valeurs qu'il prend lorsque ses variables booléennes parcourent l'ensemble $\{0, 1\}$. Comme nous mesurons la complexité de calcul d'une fonction-polynôme par la complexité minimale d'un polynôme dont elle provient, les questions qu'on se posera à propos des fonctions-polynômes seront toujours équivalentes aux questions correspondantes concernant les polynômes.

La complexité c du circuit C est par définition son nombre de portes d'addition et de multiplication ; son nombre de flèches est $2.c$, si bien qu'il n'a pas plus de $c + 1$ entrées, puisque chacune de ses portes, sauf sa sortie, émet au moins une flèche.

La profondeur p de C est la longueur maximale d'un chemin joignant une entrée à la sortie ; elle est inférieure à c, qui est elle-même majorée par 2^p. Son ampleur est le nombre entier calculé par le circuit obtenu à partir de C en étiquetant toutes ses entrées par la constante $1 = (-1)^2$; l'ampleur majore la somme des valeurs absolues des coefficients des monômes du polynôme calculé par C ; elle est inférieure à $2 \wedge 2^p \leq 2 \wedge 2^c$.

Le degré de C est calculé par le circuit obtenu à partir de C en étiquetant ses entrées par 1 (qu'elles soient variables ou constantes), en remplaçant ses multiplications par des additions, et ses additions par des prises de maximum ; le degré est majoré par 2^p, lui-même inférieur à 2^c. Le sous-degré de C est défini de la même manière que son degré, sauf qu'une entrée étiquetée par une constante se voit attribuer le sous-degré 0. On notera bien que les variables booléennes sont prises en compte comme les autres dans l'évaluation du degré et du sous-degré.

Autrement dit, le degré de C est celui du polynôme calculé par le circuit obtenu en remplaçant la constante -1 par une variable ; pour définir le sous-degré de cette manière, on remplace cette constante -1 par la constante 1.

Il est clair que le sous-degré de C majore le degré (total) du polynôme qu'il calcule. Réciproquement, si un polynôme de degré d est calculé par un circuit de complexité c, le calcul de ses parties homogènes permet de l'exprimer par un circuit de complexité $c.(d + 1)^2$ et de sous-degré d.

Le remplacement du sous-degré par le degré est par contre plus acrobatique. Si dans un circuit de sous-degré d et de complexité c on remplace chaque porte de sous-degré nul par une entrée étiquetée par l'entier relatif qu'elle calcule, on obtient un circuit de complexité moindre et de degré inférieur à $c.d + 1$ (voir [Malod, 2003]). Mais cette manipulation suppose qu'on emploie librement des constantes entières, d'ampleur doublement exponentielle, puisque borner le sous-degré ne bride pas le calcul des constantes, ce à quoi nous nous refusons ici, bien que le don gratuit de constantes arbitraires soit communément admis chez les disciples de Valiant.

La classe de complexité chérie par ce dernier est celle où l'on borne polynomialement à la fois la complexité et le degré du circuit, et pour nous ce sera bien son degré, et non pas son sous-degré. Guillaume Malod a compris tout l'avantage qu'avait la possibilité de définir cette classe en bornant polynomialement un seul paramètre du circuit, sa complexité, à condition d'imposer une condition structurelle à son architecture. Il a défini la notion de circuit multiplicativement disjoint, pour lequel chaque porte de multiplication reçoit ses flèches de deux sous-circuits disjoints. Un circuit multiplicativement disjoint de complexité c a son degré majoré par $c + 1$, tandis que son ampleur est inférieure à 2^c. Réciproquement, un circuit de complexité c et de degré d peut se remplacer par un circuit multiplicativement disjoint de complexité $c.d$, qui calcule le même polynôme (voir [Malod, 2003], [Malod and Portier, 2006]).

Enfin, les termes sont les circuits qui sont à la fois additivement et multiplicativement disjoints ; le lemme de parallélisation de [Muller and Preparata, 1976] établit l'équivalence des circuits de profondeur logarithmique et des termes de complexité polynomiale.

Tout cela mène à la définition des classes de suites de fonctions-polynômes suivantes, qui chacune ne dépend que d'un nombre polynomial de variables :

- VPdl, l'ensemble des suites f_n de fonctions-polynômes calculées par une suite C_n de circuits arithmétiques dont la complexité est majorée par un polynôme en n
- VPdb, l'ensemble des suites f_n de fonctions-polynômes calculées par une suite C_n de circuits arithmétiques dont la complexité est majorée par un polynôme en n, ainsi que leur sous-degré (ou, de façon équivalente, le degré des polynômes f_n)
- VPmd, l'ensemble des suites f_n de fonctions-polynômes calculées par une suite C_n de circuits arithmétiques multiplicativement disjoints dont la complexité est majorée par un polynôme en n (ou, de façon équivalente, par une suite de circuits arithmétiques de complexité et de degré—pas de sous-degré—polynomialement bornés)
- VPt, l'ensemble des suites f_n de fonctions-polynômes calculées par une suite C_n de termes arithmétiques dont la complexité est majorée par un polynôme en n (ou, de façon équivalente, par une suite de circuits arithmétiques de profondeur logarithmique).

VPt est incluse dans VPmd, mais on ne sait pas si cette inclusion est stricte ; il s'agit de la variante polynomiale du célèbre problème de parallélisation des algorithmes ($P =? NC_1$) ; toutefois la classe VPmd se parallélise en profondeur polylogarithmique, d'après [Valiant, Skyumi, Berkowitz, and Rackoff, 1983]. Par contre VPdl contient strictement VPmd ; en effet, le degré et le logarithme de l'ampleur d'une suite dans VPdl peuvent être exponentiels, tandis que dans VPmd ils restent polynomiaux.

Quant à la classe VPdb, qui est strictement comprise entre VPdl et VPmd pour des raisons de degré et d'ampleur, elle n'a pas beaucoup d'intérêt.

La lettre V rend hommage à Leslie Valiant, et indique que les objets calculés sont des polynômes, et non pas des fonctions booléennes ; P est pour (complexité) polynomiale, t pour terme, md pour multiplicativement disjoint, db pour degré borné et dl pour degré libre. A strictement parler, ces classes ont été introduite

par [Malod, 2003], qui utilise des notations un peu différentes ; en particulier il emploie le signe ° pour indiquer que -1 est la seule constante utilisée, ce dont nous nous abstenons puisque chez nous c'est une convention systématique. Les travaux originaux de Valiant ont pour cadre une classe définie de façon semblable à VPdb, à ceci près qu'est autorisé l'emploi gratuit de constantes rationnelles (dans ce cas, comme les constantes autorisées forment un anneau, $\mathrm{VPdb}(Q) = \mathrm{VPmd}(Q)$).

Nous définissons maintenant les sommations de Valiant. Soit $f(\underline{u}, \underline{v}, \underline{x})$ une fonction-polynôme, dont $\underline{u}$ et $\underline{v}$ forment les variables booléennes, et $\underline{x}$ les variables non-booléennes. Nous appelons sommation de Valiant l'opération $\Sigma_{\underline{v} \in \{0,1\}^{\wedge n}} f(\underline{u}, \underline{v}, \underline{x})$, dont le résultat est une fonction-polynôme qui ne dépend plus du uplet de variables booléennes $\underline{v}$. Si $\underline{v}$ est vide, on convient que l'opération est neutre, $\Sigma_{\varnothing} f(\underline{u}, \underline{x}) = f(\underline{u}, \underline{x})$.

En appliquant une sommation de Valiant devant chaque point des suites appartenant aux classes VPt, VPmd, VPdl, on obtient respectivement les classes VNPt, VNPmd, VNPdl. Il est connu que, sous une sommation de Valiant, circuits multiplicativement disjoints et termes sont équivalents, c'est-à-dire que $\mathrm{VNPt} = \mathrm{VNPmd}$; c'est une étape essentielle de la preuve du célèbre théorème de Valiant sur la complétude du permanent pour la classe VNPmd (qui demande l'emploi de la constante $1/2$; voir [Valiant, 1979]) ; on en trouvera une démonstration très conceptuelle dans [Malod, 2003].

Par contre, on ne sait pas séparer VNPmd de VPmd : c'est la question fondamentale posée par Valiant (c'est en fait une variante un peu plus stricte de cette question, puisque nous n'autorisons que -1 comme constante) ; on ne sait pas d'avantage séparer VNPdl de VPdl.

Les égalités $\mathrm{VNPmd} = \mathrm{VPmd}$ et $\mathrm{VNPdl} = \mathrm{VPdl}$, ou même la seule inclusion de VNPmd dans VPdl, semblent toutefois bien improbables, car elles impliquent $\mathrm{P} = \#\mathrm{P}$, et en conséquence $\mathrm{P} = \mathrm{NP}$ (ou plus exactement la version non-uniforme de ces assertions, puisqu'il n'est pas dans l'usage d'introduire de conditions d'uniformité dans la définition des classes de complexité de Valiant). Pour le voir, on utilise tout d'abord la réduction parcimonieuse des circuits booléens aux termes booléens (voir les démonstrations des Lemmes 6 et 10) ; on mime ensuite par des termes arithmétiques une quelconque suite $f_n(\underline{u})$ de termes booléens de complexité polynomiale, et on observe que leur somme de Valiant exprime le nombre de solutions de l'équation $f_n(\underline{u}) = 1$; comme ce nombre est simplement exponentiel, son calcul dans VPdl se remplace par un calcul en chiffres dans P.

La lettre N est pour non-déterminisme ; comme l'ont remarqué plusieurs auteurs, ce choix n'est ici pas très heureux, à admettre qu'il le soit dans le cas de la complexité classique, booléenne ; en effet, il n'y a pas de vraie analogie de comportement entre la classe NP et les classes VNP, qui sont à rapprocher plutôt de la classe #P. En particulier, si on borne le degré, les sommations ne produisent pas de hiérarchie, comme le montre le Théorème 2 qui suit. Après réflexion, nous n'avons pas changé cette notation établie par l'usage, d'autant plus qu'il y a un parallèle entre les classes VNP et NP qui sera exposé dans la dernière section de cet article.

REMARQUE 1. Les classes de complexité booléennes non-uniformes sont usuellement munies du suffixe /poly, que n'emploie pas l'auteur de ces lignes pour protester contre la démarche absurde qui consiste à introduire de l'uniformité par un dispositif combinatoire aussi compliqué qu'une machine de Turing pour ensuite la détruire

par un artifice ; dans tous les résultats exposés ici, l'uniformité n'aurait qu'un effet décoratif parasite, sans rapport avec le fond du problème. Il trouve par ailleurs incohérent de mettre de l'uniformité aux classes booléennes et pas à celles de Valiant, et s'engage à attacher du /poly aux classes booléennes quand les fétichistes de l'effectivité en accrocheront aussi aux classes de Valiant.

§2. Les circuits S-arithmétiques. Dans la définition des classes VNP la sommation de Valiant n'est appliquée qu'une fois, à la sortie du circuit. Nous introduisons maintenant des sommations partout dans les calculs, et pour cela nous considérons des circuits que nous qualifions de S-arithmétiques (S = sommations ; nous dirons aussi S-circuits) et qui, en plus des portes d'additions et de multiplication, comportent des portes de sommation, qui ne reçoivent qu'une seule flèche, et effectuent une sommation relative à un certain uplet $\underline{v}$ de variables booléennes.

La complexité du circuit est par définition son nombre de portes d'addition, de multiplication et de sommation ; son nombre d'entrées est majoré par la somme de son nombre de portes d'addition et de son nombre de portes de multiplication, augmentée d'une unité.

L'ensemble des variables libres associées à une porte p d'un S-circuit est défini de manière inductive et obvie ; si p est une entrée indexée par -1, cet ensemble est vide ; si p est une entrée indexée par une variable, cet ensemble est le singleton de cette variable ; si p est une addition, $p = p' + p''$, ou bien une multiplication, $p = p'.p''$, cet ensemble est la réunion de l'ensemble des variables libres de p' et de celui de p'' ; si p est une sommation $\Sigma_{\underline{v}} p'$, son ensemble de variables libres est celui de p' privé de $\underline{v}$.

Nous exigeons qu'une sommation porte sur un uplet de variables libres à la porte p' ; cette exigence n'est pas très forte, car en trois opérations on peut ajouter puis retrancher une nouvelle variable v à la fonction-polynôme calculée en p' ; elle évite d'avoir à tenir compte, dans l'évaluation de la complexité, du nombre de variables de sommation qui n'apparaîtraient pas aux entrées du circuit. Une autre façon de faire serait de n'introduire que des sommations portant sur une seule variable ; une sommation sur n variables, devant être décomposée en n sommations successives portant sur une variable, se verrait alors attribuer la complexité n.

Une suite de fonctions-polynômes est dite dans VSPdl si elle est calculée par une suite de circuits S-arithmétiques de complexité polynomiale ; si on contraint ces circuits à être multiplicativement disjoints, ou bien à être des termes, on obtient respectivement les classes VSPmd et VSPt.

Comme une sommation n'affecte pas le degré et multiplie l'ampleur par un nombre simplement exponentiel, il n'y a pas de séparation évidente entre une classe VSP et la classe VNP correspondante. Cependant, ces sommations dispersées dans le circuit n'apportent rien de nouveau si on borne le degré, car nous allons voir que, si ce circuit est multiplicativement disjoint, on peut faire remonter toutes les sommations en une seule, effectuée à la sortie, au prix d'une augmentation polynomiale de la complexité ; autrement dit VSPmd = VNPmd(= VSPt = VNPt).

Dans un S-circuit, une même variable peut avoir à la fois des occurrences libres et des occurences liées : c'est une chose qu'il est impossible de corriger radicalement si le circuit n'est pas un terme. L'inconvénient, c'est que si v est liée dans le calcul de $f(u, v)$, on ne calcule pas $f(u, 1)$ en remplaçant v par $1 = (-1)^2$ aux entrées du

circuit ; de même, on n'obtient pas $f(w, w)$ en remplaçant u et v par w aux entrées du circuit. Mais par bonheur, une sommation de Valiant permet de faire, pour un faible coût, un changement de variables booléennes non pas aux entrées, mais à la sortie du circuit, grâce aux formules $f(u, 1) = \Sigma_v v.f(u, v)$, $f(u, 0) = \Sigma_v(1 - v).f(u, v)$, $f(w, w) = \Sigma_{u,v}((u - w)^2 - 1).((v - w)^2 - 1).f(u, v)$. Ce coût est effectivement très faible car si, après avoir calculé $g(u)$, on veut k exemplaires supplémentaires $g(v_1), \ldots, g(v_k)$ de cette fonction, la formule $g(v) = \Sigma_u(-1).((v + (-1).u)^2 + (-1)).g(u)$ permet de les obtenir avec une complexité qui augmente de $7k$, au lieu d'être multipliée par $k + 1$.

Nous avons adopté la convention de distinguer des variables booléennes des autres, à qui il est interdit d'apparaître sous une sommation ; elle nous permet de montrer facilement que la classe VSPdl est close par composition, les remplacements de variables booléennes par des fonctions booléennes pouvant se faire à la sortie au lieu de se faire à l'entrée : si les variables x et y pouvaient jouer un rôle booléen dans les sommations et un rôle non-booléen par ailleurs, on ne pourrait déduire un calcul de $f(z, z)$ à partir d'un calcul de $f(x, y)$ ni par remplacement aux entrées, ni par changement de variable à la sortie !

Cependant, cette convention n'est pas coûteuse. Si on ne la fait pas, c'est-à-dire si on tolère qu'une variable x puisse apparaître sous une sommation $\Sigma_{x \in \{0,1\}}$ et prenne ailleurs des valeurs arbitaires, pour s'y ramener il suffit de remplacer aux entrées x par $u.x + v$, où u et v sont des variables booléennes ; on remplace $\Sigma_x f(x, \underline{y})$ par $\Sigma_{u,v}(1 - u).f(u.x + v, \underline{y})$, et on se débarasse de u et de v à la sortie grâce à une sommation $\Sigma_{u,v} u.(1 - v).f(u.x + v, \underline{y})$. Cette manipulation n'a qu'un effet bénin sur la complexité des circuits, et ne change pas la définition de la classe VSPdl. Comme elle n'affecte pas la disjonction des multiplications, elle conserve également la classe VSPmd et l'égalité VSPmd = VNPmd.

Ce pouvoir miraculeux qu'ont les sommations de Valiant de dupliquer gratuitement, ou presque, les fonctions de variables booléennes va être utilisé si souvent que nous évaluons dès à présent le nombre d'opérations nécessaires pour changer le nom de n variables booléennes dans une fonction. La formule est $f(v_1, \ldots, v_n) = \Sigma_{\underline{u}}(-1)^n.((u_1 - v_1)^2 - 1). \ldots .((u_n - v_n)^2 - 1).f(u_1, \ldots, u_n)$; chacun des facteurs $(u_i + (-1).v_i)^2 + (-1)$ demande quatre opérations ; il faut ensuite effectuer leur produit avec $f(\underline{u})$, ce qui demande n multiplications, multiplier éventuellement par -1 et faire la sommation ; cela fait en tout $5n + 1$ ou $5n + 2$ opérations, suivant la parité de n.

Le changement de variables booléennes nous permet de nous ramener à des S-circuits satisfaisant une contrainte particulière. Nous dirons que le S-circuit C est régulier si, à chacune de ses portes p, une variable u qui est libre en p n'apparaît jamais comme variable de sommation dans le sous-circuit Cp. Autrement dit, relativement à une porte p d'un circuit régulier, les variables se répartissent en trois groupes disjoints : les variables qui sont liées dans le circuit Cp ; celles qui sont libres en p ; celles qui ne figurent pas aux entrées de Cp.

La régularité sera utilisée dans la démonstration du Théorème 2 pour faire sauter les sommations par dessus les additions ; supposons qu'on sache calculer $f(u)$, et que nous voulions calculer $f(u) + f(0) + f(1) = f(u) + \Sigma_u f(u)$ en peu d'opérations supplémentaires, mais en mettant la sommation à la sortie du circuit ; comme u est à

la fois libre et liée à la sortie, nous devons d'abord changer cette variable booléenne en une autre, v, grâce à la formule $f(v) = \Sigma_u(1 - (u - v)^2).f(u)$, et finalement $f(v) + f(0) + f(1) = \Sigma_u(2 - (u - v)^2).f(u)$, ce qui n'ajoute que huit opérations, dont la sommation de sortie, au calcul de $f(u)$. Pour calculer $f(u) + f(0) + f(1)$, il n'y avait qu'à prévoir la bonne variable au départ ! Aucune formule de ce genre ne permet de calculer $f(u).(f(0) + f(1))$: nous ne ferons sauter les sommations par dessus les produits que dans les circuits *multiplicativement disjoints*, grâce au Théorème de Fubini, qui demande que les ensembles de variables de sommation des deux facteurs soient disjoints.

LEMME 1, DE RÉGULARISATION. *Un* S*-circuit* C, *de complexité* c, *peut se remplacer par un* S*-circuit régulier* C^*, *de complexité* $10.c^2$, *qui calcule la même fonction-polynôme. Si* C *est multiplicativement disjoint, on peut obtenir* C^* *multiplicativement disjoint de complexité* $14.c^2$.

DÉMONSTRATION. Pour chaque porte p calculant la fonction $p(\underline{u})$, le circuit C^* contient une porte p^* calculant la même fonction, mais dans un jeu de variables $\underline{u}^*$ différent, et qui lui est propre ; nous ne mentionnons que les variables booléennes, car on ne touche pas aux autres, et nous notons n leur nombre.

La construction de C^* se fait par induction sur la profondeur du sous-circuit Cp.

On ne change rien aux entrées.

Si p est une porte de sommation, $p(\underline{u}) = \Sigma_{\underline{v}} p'(\underline{u}, \underline{v})$, la porte p'^* déjà construite calcule $p'(\underline{u}^*, \underline{v}^*)$ dans un nouveau jeu de variables ; il nous faut effectuer la sommation sur $\underline{v}^*$, et changer le nom des variables survivantes de $\underline{u}^*$, ce qui se fait en une seule sommation et au plus $5n + 2$ opérations.

Si p est une porte d'addition ou de multiplication, $p = p' + p''$ ou $p = p'.p''$, il nous faut d'abord exprimer les fonctions calculées en p'^* et p''^* dans un nouveau jeu de variables avant de les additionner ou de les multiplier, ce qui demande au plus $2.(5n + 2) + 1 = 10.n + 5$ opérations.

Comme nous avons introduit des variables nouvelles à chaque porte p^*, C^* est régulier ; pour qu'il calcule la fonction demandée, il n'y a qu'a prévoir que ses variables de sortie soient les bonnes.

Le nombre de variables est majoré par le nombre d'entrées, qui l'est par $c + 1$; mais si le circuit C a $c + 1$ entrées, il ne comporte pas de sommation et le résultat est évident ; si le circuit a c entrées et comporte une sommation, c'est un terme, et pour le régulariser il suffit de changer le nom des variables de sommation à ses entrées. Et s'il n'y pas plus de $c - 1$ entrées, la complexité de C^* est majorée par $c.(10.(c - 1) + 5) = 10.c^2 - 5.c \leq 10.c^2$.

Si C est multiplicativement disjoint, et si on veut que C^* le reste, il faut effectuer les multiplications auxiliaires de manière disjointe, c'est-à-dire calculer les expressions $(u_i - v_i)^2 = (u_i - v_i).(u_i - v_i)$ en cinq opérations au lieu de trois, ce qui ajoute $2.n$ à la complexité des changements de variables. La complexité de C^* est alors bornée par $c.(14.(c - 1) + 5) \leq 14.c^2$. ⊣

Ce lemme et le théorème suivant montrent que VSPmd = VNPmd.

THÉORÈME 2. *Si* C *est un circuit* S*-arithmétique multiplicativement disjoint et régulier de complexité* c, *on peut construire un circuit arithmétique multiplicativement disjoint de complexité* $7.c^2$, *calculant une fonction-polynôme* $f(\underline{u}, \underline{v}, \underline{x})$ *dont la sommation de Valiant* $\Sigma_{\underline{v}} f(\underline{u}, \underline{v}, \underline{x})$ *est la fonction-polynôme calculée par* C.

DÉMONSTRATION. On suppose que $c \geq 2$, car le théorème est évident pour $c = 0$ et pour $c = 1$.

L'idée est d'utiliser le Théorème de Fubini, qui déclare que $(\Sigma_{\underline{u}} f(\underline{u})).(\Sigma_{\underline{v}} f(\underline{v})) = \Sigma_{\underline{u},\underline{v}} f(\underline{u}).f(\underline{v})$ quand les uplets $\underline{u}$ et $\underline{v}$ sont disjoints. Cependant, comme une même variable peut étiqueter des entrées différentes, le fait que le circuit soit multiplicativement disjoint n'implique pas qu'à une porte de multiplication les uplets de variables de sommation soient disjoints. Nous remédions à cela en introduisant un nouveau uplet $\underline{y}$ de variables—une variable par entrée étiquetée par une variable—et nous notons s la surjection canonique de $\underline{y}$ sur les variables de C ; $s(y') = s(y'')$ signifie donc que les entrées indexées par y' et par y'' sont étiquetées dans C par la même variable.

Nous construisons, par induction sur la profondeur, un circuit arithmétique multiplicativement dispoint C*, comprenant, pour chaque porte p de C, une porte p^* calculant $p^*(\underline{u}, \underline{z})$, où $\underline{u}$ est l'ensemble des variables de $\underline{y}$ dont l'image par s est liée à la porte p, et $\underline{z}$ est l'ensemble des variables de $\underline{y}$ dont l'image par s est libre à la porte p, telle que $\Sigma_{\underline{u}} p^*(\underline{u}, s(\underline{z}))$ soit la fonction-polynôme calculée dans C à la porte p.

Si p est une entrée indexée par -1, p^* aussi ; si p est une entrée indexée par $s(y)$, p^* est une entrée indexée par y.

Si p est une porte de produit, $p = p'.p''$, les fonctions $p'^*(\underline{u}', \underline{z}')$ et $p''^*(\underline{u}'', \underline{z}'')$ n'ont pas de variables en commun puisque C est multiplicativement disjoint, et p^* est tout simplement le produit de p'^* et de p''^*. Ça reste multiplicativement disjoint.

Si p est une porte d'addition, $p = p' + p''$, la régularité de C impose qu'aucune variable de $\underline{u}'$ n'apparaisse dans $\underline{z}''$, ni aucune variable de $\underline{u}''$ dans $\underline{z}'$. Si nous notons $\underline{u}$ la réunion de $\underline{u}'$ et de $\underline{u}''$, et $\underline{z}$ celle de $\underline{z}'$ et de $\underline{z}''$, nous posons $p^*(\underline{u}, \underline{z}) = U'.p'^*(\underline{u}', \underline{z}') + U''.p''^*(\underline{u}'', \underline{z}'')$, où U' est le produit des variables de $\underline{u}''$ qui ne sont pas dans $\underline{u}'$, et U'' celui des variables de $\underline{u}'$ qui ne sont pas dans $\underline{u}''$; ces facteurs servent à éviter qu'après sommation $p'^*(\underline{u}', \underline{z}')$ et $p''^*(\underline{u}'', \underline{z}'')$ soient multipliés par des puissances intempestives de 2. S'il n'y a pas plus de n variables, ce calcul ne demande pas plus de $n + 1$ opérations, avec multiplications disjointes.

Supposons maintenant que p soit une porte de sommation, recevant sa flèche de p' ; la fonction-polynôme calculée en p'^* dépend des variables $\underline{u}$, $\underline{v}$ et $\underline{z}$, où les variables de $\underline{v}$ sont celles dont l'image par s intervient dans la sommation en p. Puisque C est régulier, s ne peut identifier une variable de $\underline{u}$ et une variable de $\underline{v}$. La fonction-polynôme $p^*(\underline{u}, \underline{v}, \underline{z})$ s'obtient en faisant passer $\underline{v}$ du côté de $\underline{u}$, à ceci près qu'il faut restreindre la sommation aux $\underline{v}$ tels que $v_1 = v_2$ si $s(v_1) = s(v_2)$; pour ce faire on multiplie $p'^*(\underline{u}, \underline{v}, \underline{z})$ par des facteurs du type $(v_1 - v_2)^2 - 1$, qui est nul si $v_1 \neq v_2$; comme on doit disjoindre les multiplications, chacun de ces facteurs demande 6 opérations, et en jouant sur la transitivité de l'égalité on n'a pas besoin d'en mettre plus de $n - 1$; on a peut-être besoin d'une multiplication par -1 pour que le résultat soit 1, et non pas -1, quand toutes les égalités sont vérifiées, si bien que pour effectuer ce produit il ne faut pas plus de $7.(n-1)+1 = 7.n-6$ opérations. Comme $n \geq 2$, $7.n - 6 > n + 1$.

Si le circuit C a $c + 1$ entrées, il n'a pas de portes de sommation, et $C^* = C$ (au changement de variables près !). Sinon, n est majoré par c, et la complexité de C* par $c.(7.c - 6) = 7.c^2 - 6.c < 7.c^2$.

Pour obtenir le circuit cherché, on remplace les variables libres à la sortie de C* par leurs images par s. ⊣

§3. La fonction-coefficient. La fonction-coefficient du polynôme $f(x_1, \ldots, x_n)$ est ainsi définie : si $\underline{v}_1, \ldots, \underline{v}_n$ sont les écritures en base deux des entiers $d_1, \ldots, d_n$, $cf(\underline{v}_1, \ldots, \underline{v}_n)$ est le coefficient dans f du monôme $x_1 \wedge d_1 \ldots . x_n \wedge d_n$. On peut aussi définir des fonctions-coefficients partielles, en considérant f comme un polynôme en un sous-ensemble de l'ensemble de ses variables, la fonction-coefficient dépendant alors des autres variables ; c'est ainsi que, pour une fonction-polynôme, on considère en général sa fonction-coefficient par rapport à ses variables non-booléennes.

Nous considérons également le polynôme réduit de f, qui est obtenu de la manière suivante. Si x est une variable, $x^d = x \wedge (v_0 + 2.v_1 + \cdots + 2^{m-1}.v_{m-1}) = x^{v0}.(x^2)^{v1} \ldots . (x \wedge 2^{m-1})^{vm-1}$; nous introduisons, pour chacune de ces variables x (non-booléenne) un m-uplet de nouvelles variables $y_0, \ldots, y_{m-1}$, et dans l'expression de chaque monôme nous remplaçons $x \wedge 2^i$ par y_i. Ce polynôme réduit a même fonction-coefficient que f ; il est de degré un par rapport à chacune de ses variables non-booléennes, et f s'obtient à partir de son polynôme réduit en y remplaçant des variables par des puissances de variables.

Dans chacune des classes de complexité que nous considérons, le nombre de variables est polynomial et le degré au pire simplement exponentiel, si bien que les fonctions-coefficients ne dépendent que d'un nombre polynomial d'arguments, et qu'il est raisonnable de chercher à évaluer leur complexité.

Si $\underline{v} = (v_0, \ldots, v_{m-1})$, nous notons $x^{\underline{v}}$ le produit des $v_i.(x \wedge 2^i - 1) + 1$; ce polynôme se calcule en $5.m - 1$ additions et multiplications ($1 = (-1)^2$ n'est calculé qu'une fois !) ; si $\underline{v}$ prend une valeur booléenne, $x^{\underline{v}}$ est la puissance de x par l'entier d dont $\underline{v}$ est le développement binaire. Comme un polynôme est somme de ses monômes, $f(\underline{x}) = \Sigma_{\underline{v}} cf(\underline{v}, \ldots, \underline{v}_n).x_1 \wedge \underline{v}_1 \ldots . x_n \wedge \underline{v}_n$, si bien que, si la suite cf_s est dans VNPdl, il en est de même de la suite f_s. La même chose vaut pour la classe VNPmd, car alors le degré est petit, et le circuit calculant $x^{\underline{v}}$ est de profondeur logarithmique.

Malod a montré que, réciproquement, la classe VNPt = VNPmd est close pour la prise de fonctions-coefficients (et en conséquence la prise de polynômes réduits !), et qu'en fait l'hypothèse VNPmd = VPmd équivaut à la clôture de VPmd pour la prise de fonction-coefficient ([Malod, 2003]) ; il a été précédé en cela par un résultat de [Valiant, 1982], qui ne peut parler que du coefficient d'un monôme isolé dans un développement partiel d'un polynôme, faute d'avoir introduit des variables booléennes.

On ne sait pas si la classe VNPdl est close pour la prise de fonctions-coefficients. Par contre, c'est le cas pour la classe VSPdl, comme le montre le théorème suivant :

THÉORÈME 3. *Soit* C *un* S*-circuit de complexité* $c > 0$, *ayant* n *variables non-booléennes dont le degré de chacune s'écrit avec* m *chiffres. Sa fonction-coefficient peut se calculer par un* S*-circuit de complexité* $22.c.m.n < 22.(c+1)^3$, *n'ayant pas plus de* $2.c$ *portes de sommation.*

DÉMONSTRATION. Nous allons construire, par induction sur la profondeur, un S-circuit C* contenant, pour chaque porte d'opération p de C, une porte p^* calculant la fonction-coefficient du polynôme calculé en p ; nous rappelons que ce polynôme est considéré comme développé en monômes par rapport à ses variables non-booléennes seulement, les variables booléennes intervenant dans l'expression des coefficients de ces monômes. Pour exprimer la fonction-coefficient, il faut $n.m$

variables booléennes supplémentaires, mais ce jeu de variables dépendra de p^* : il ne sera pas fixe, et même nous en changerons souvent.

Nous rappelons que le changement de $n.m$ variables booléennes dans une fonction demande $5.n.m + 2$ opérations, dont une sommation.

Il nous faudra simuler polynomialement l'addition de deux nombres de m chiffres, écrits en base deux. Si on additionne deux nombres x et y de un chiffre, on obtient un nombre de deux chiffres, rt, où t est la somme modulo deux de x et de y, et où r est la retenue ; $r = x.y$, $t = x + y - 2.xy$, ce qui se calcule en 4 opérations (sommes et produits) si on prévoit d'ajouter au circuit une porte calculant $-2 = (-1)+(-1)$.

Si on additionne trois nombres de un chiffres x, y et z, on obtient également un nombre de deux chiffres rt ; la retenue est la fonction-majorité de x, y et z, qui vaut $r = xy + yz + zx - 2.xyz$; la somme modulo deux vaut $t = x + y + z - 2.(xy + yz + zx) + 4.xyz$; leur calcul demande 12 opérations (qui dit mieux ?), à condition de prévoir une porte de plus calculant $4 = (-2)^2$.

Nous ne ferons ces additions de nombres de m chiffres que dans la situation où la dernière retenue est nulle, et n'a pas besoin d'être calculée ; une telle addition en chiffre demande donc $4 + 12.(m-2) + 10 = 12.m - 10$ additions et multiplications, sans compter les deux portes calculant les constantes -2 et 4 qu'il ne faudra pas oublier d'ajouter.

Cela étant dit, nous construisons C* :

PREMIER CAS. p est une porte d'addition, recevant ses flèches de p' et de p''.

Supposons d'abord que p' et p'' soient toutes deux des portes d'opération ; si p'^* et p''^* expriment leurs fonctions-coefficients dans le même jeu de variables (par exemple si $p' = p''$), il suffit de les additionner ; sinon, avant de ce faire, on commence par remplacer le jeu de variables de p''^* par celui de p'^* ; tout ceci demande au plus $5.nm + 3$ opérations.

Supposons maintenant que p' soit une porte d'opération et p'' une entrée étiquetée par la constante -1. On gardera le même jeu de variable v_{ij}, $1 \leq i \leq n$, $0 \leq j \leq m-1$, que pour la fonction calculée en p'^*, et on modifiera son coefficient constant en lui ajoutant $(-1).(-1)^{mn}.\Pi(v_{ij}-1)$, ce qui demande $2.nm$ ou $2.nm+1$ opérations.

Si p'' est étiquetée par une variable booléenne, on ajoute cette dernière au coefficent constant, au lieu de lui ajouter -1, en $2.nm + 2$ opérations.

Si p' est une porte d'opération et p'' une entrée indexée par la variable x_k non booléenne, on garde le même jeu de variables et on augmente d'une unité le degré du monôme x_k en ajoutant $v_{k1}.(-1)^{nm-1}.\Pi_{ij \neq k1}(v_{ij} - 1)$, ce qui ne demande pas plus de $2.nm$ opérations.

Enfin, si p' et p'' sont deux entrées, le polynôme calculé par p est de la forme $a + b$, $x + a$, $2.x$ ou $x + y$, où a et b valent -1 ou une variable booléenne ; il n'a pas plus de deux monômes ; comme la fonction caractéristique d'un nm-uplet booléen (celle qui vaut 1 pour ce uplet et 0 pour les autres) se calcule en $2.nm$ opérations, sa fonction-coefficient se calcule en au plus $4.nm + 3$ opérations.

La contribution maximale d'une porte d'addition à la construction de C* est donc de $5.nm + 3$ portes, dont une porte de sommation.

DEUZIÈME CAS. p est une porte de sommation, recevant sa flèche de p' ; si cette dernière est une entrée, le polynôme calculé en p est de la forme -1, 1, u ou x, et la fonction-coefficient de ce monôme s'exprime en $2.nm + 1$ opérations.

Si p' est une porte d'opération, p^* fait la même sommation que p.

TROISIÈME CAS. p est une porte de multiplication, recevant ses flèches de p' et p''.

Si p' et p'' sont des entrées, p calcule un monôme de la forme ab, ax, x^2 ou xy, dont la fonction-coefficient se calcule en $2.nm + 2$ opérations.

Si p' est une porte d'opération et p'' une entrée étiquetée par -1, on se contente de multiplier par -1 la fonction calculée en p'^*. De même, si p'' est étiquetée par une variable booléenne, on multiplie par cette variable.

Si p' est une porte d'opération et p'' une entrée étiquetée par la variable x_k, il faut augmenter d'une unité les degrés en cette variable ; pour cela, on introduit de nouvelles variables $\underline{u}_k$ pour remplacer les variables $\underline{v}_k$ exprimant la puissance de x_k et on calcule $\Sigma_{\{\underline{v}k/\underline{v}k+1=\underline{u}k\}} p'^*(\ldots \underline{v}_k, \ldots)$, l'expression $\underline{v}_k + 1 = \underline{u}_k$ signifiant que si on ajoute 1 au nombre exprimé en binaire par $\underline{v}_k$ on obtient le nombre exprimé par $\underline{u}_k$. Pour restreindre la sommation à cet ensemble, nous devons d'abord simuler l'addition de 1 à un nombre de m chiffres, ce qui se fait en $4.m$ opérations (on additionne m fois deux nombres de un chiffre), puis effectuer le remplacement des m coordonnées de $\underline{v}_k + 1$ par celles de $\underline{u}_k$, ce qui demande $5.m + 2$ opérations. La contribution de la porte p est alors de $9.m + 2$ opérations, dont une sommation.

Reste le cas où p' et p'' sont deux portes d'opérations ; nous commençons par exprimer la fonction calculée en p''^* en un jeu de variables fraîches $\underline{w}$, disjoint de celui qui est utilisé en p'^*, ce qui demande $5.nm + 2$ opérations ; il nous faut ensuite calculer le produit de convolution $p^*(\underline{u}) = \Sigma_{\{\underline{v},\underline{w}/\underline{v}+\underline{w}=\underline{u}\}} p'^*(\underline{v}).p''^*(\underline{w})$, la sommation étant restreinte aux $(\underline{v}, \underline{w})$ tels que la somme du n-uplet d'entiers représenté par $\underline{v}$ et du n-uplet d'entiers représenté par $\underline{w}$ donne celui représenté par $\underline{u}$. Pour cela, il nous faut d'abord effectuer les n additions en chiffres au prix de $12.nm - 10.n$ opérations, faire le produit de p'^* et de p''^*, et enfin changer les coordonnées de $\underline{v} + \underline{w}$ en celles de $\underline{u}$ en $5.nm + 2$ opérations.

La contribution de cette porte de multiplication est donc de $22.nm - 10.n + 5$ opérations, dont deux sommations, ce qui est le maximum de ce que nous avons obtenu puisque nous pouvons supposer que $nm \geq 1$, le cas $n = 0$ étant trivial.

Le S-circuit C^* ne comprend pas plus de $2.c$ portes de sommation, et sa complexité est majorée par $c.(22.nm - 10.n + 5) + 2$, les deux dernières portes servant à calculer les constantes -2 et 4. Cette quantité est inférieure à $22.c.n.m$, et on obtient la dernière majoration en majorant n par $c + 1$ et m par $c + 2$. ⊣

REMARQUE 2. Il est facile d'adapter la démonstration du Théorème 3 pour en obtenir une du résultat de Malod déclarant que la classe VNPmd est close pour la prise de fonction-coefficient.

§4. Pascaline et Factorielle. Nous appelons $\text{Pascaline}_n(\underline{u}, \underline{v})$ la fonction numérique qui au deux n-uplets booléens $\underline{u}$ et $\underline{v}$ de longueur n associe le coefficient binomial $C_N{}^M = ({}^N_M)$, où N est l'entier dont le développement binaire est $\underline{u}$, tandis que M est l'entier dont le développement binaire est $\underline{v}$. Le Théorème 3 a pour conséquence que la Pascaline est dans VSPdl. En effet :

THÉORÈME 4. *Pascaline_n se calcule par un circuit* S*-arithmétique de complexité* $72.n^2$, *comprenant* $2.n$ *portes de sommation.*

DÉMONSTRATION. La Pascaline est intimement liée à la fonction-coefficient $C_n(\underline{u}, \underline{v}, \underline{w})$ du polynôme $(x + y)^{\underline{u}}$, où $\underline{u}$ est un n-uplet booléen ; $\underline{v}$ est le n-uplet booléen représentant le degré de la variable x, et $\underline{w}$ celui qui représente le degré de la variable y. Plus précisément, $\text{Pascaline}_n(\underline{u}, \underline{v}) = C_n(\underline{u}, \underline{v}, \underline{u} - \underline{v})$, où $\underline{u} - \underline{v}$ est le résultat de la soustraction en chiffres de $\underline{v}$ à $\underline{u}$ (qui vaut 0 si $\underline{u} \leq \underline{v}$).

Le polynôme $(x + y)^{\underline{u}}$ est le produit des $M_i = 1 + u_i.((x + y) \wedge 2^i - 1)$, pour $0 \leq i \leq n - 1$; il est calculable en $5.n$ opérations. En remplaçant c par $5.n$, m par n et n par 2 dans le résultat du Théorème 3, on majore la complexité de C_n par $220.n^2$. Nous allons examiner de plus près l'algorithme de calcul de C_n pour améliorer la constante.

Nous calculons tout d'abord la fonction-coefficient $c_i(\underline{v}_i, \underline{w}_i)$ de $(x + y) \wedge 2^i$, chaque fois dans un jeu de variables neuf. Pour celle de $x+y$, nous formons le produit des $(v_{0i}-1).(w_{0i}-1)$ pour $i \leq 0$, que nous multiplions par $v_{00}+w_{00}-2.v_{00}w_{00}$, ce qui demande $4.n$ opérations, compte non tenu du calcul de la constante -2 ; ensuite, pour calculer les puissances de $x + y$, on fait $n - 1$ multiplications successives, chacune contribuant pour $44.n - 15$ opérations au calcul des coefficients ; cette étape demande donc $4.n + (44.n - 15)(n - 1) + 2 = 44.n^2 - 55.n + 17$ opérations (les deux dernières portes calculent -2 et 4), dont $2.(n - 1)$ sommations.

Comme $M_i = u_i.(x+y) \wedge 2^i + (1 - u_i)$, pour obtenir la fonction-coefficient m_i de ce dernier, on multiplie c_i par u_i et on augmente son coefficient constant de $1 - u_i$ en lui ajoutant $(-1).(u_i - 1).\Pi(v_{ij} - 1).(w_{ij} - 1)$, ce qui demande $4.n + 4$ opérations. Comme il y a n fonctions m_i à calculer, ça fait en tout $4.n^2 + 4.n$ opérations à cette étape.

Nous introduisons de nouvelles variables $\underline{v}$ et $\underline{w}$ pour calculer le produit de convolution $\Sigma_{\underline{v}0+\cdots+\underline{v}n-1=\underline{v},\ \underline{w}0+\cdots+\underline{w}n-1=\underline{w}} c(\underline{v}_0, \underline{w}_0). \ldots .c_{n-1}(\underline{v}_{n-1}, \underline{w}_{n-1})$; il nous faut pour cela effectuer deux additions de n nombres de n chiffres, ce qui demande $2.(n - 1).(12.n - 10) = 24.n^2 - 44.n + 20$, les constantes -2 et 4 ayant déja été calculées. Ensuite il nous faut calculer la fonction caractéristique de l'ensemble sur lequel porte la sommation, puis effectuer cette dernière, ce qui revient à changer $2.n$ variables booléennes, soit $10.n + 1$ opérations.

Pour résumer, le calcul de la fonction C_n demande $44.n^2 - 55.n + 17 + 4.n^2 + 4.n + 24.n^2 - 44.n + 20 + 10.n + 1 = 72.n^2 - 85.n + 38$ opérations, dont $2.(n-1)+1 = 2.n-1$ sommations.

La Pascaline est donnée par la formule $\text{Pascaline}_n(\underline{u}, \underline{v}) = \Sigma_{\{\underline{w}/\underline{v}+\underline{w}=\underline{u}\}} C_n(\underline{u}, \underline{v}, \underline{w})$. Il faut $12.n - 10$ opérations pour l'addition, et $5.n + 2$ pour la fonction caractéristique et la disparition de $\underline{w}$, dont une sommation.

La Pascaline se calcule donc en $72.n^2 - 68.n + 30$ opérations, dont $2.n$ sommations de Valiant. ⊣

Nous introduisons également la fonction Factorielle_n, qui à un n-uplet booléen associe la factorielle du nombre dont il est la représentation en base deux. Il est bien connu que, si on sait calculer la Pascaline, on sait aussi calculer la Factorielle, ce qui met également cette dernière dans VSPdl ; l'idée est d'utiliser la formule $(2.N)! = C^N_{2.N}.(N!)^2$; comme la suppression du premier chiffre d'un nombre ne le divise pas exactement par deux, la formule de récurrence est la suivante : $\text{Factorielle}_{n+1}(u_0, u_1, \ldots, u_n) = \text{Pascaline}_{n+1}(0, u_1, \ldots, u_n; u_1, \ldots, u_n, 0).(1 + u_0.(2.u_1 + \cdots + 2n.u_n)).\text{Factorielle}_n(u_1, \ldots, u_n)^2$. En appliquant cette méthode,

compte tenu de notre évaluation de la complexité de la Pascaline, on obtient un circuit S-arithmétique de complexité $25.n^3$, comportant moins de $(n+1)^2$ sommations de Valiant, qui calcule Factorielle$_n$.

Nous ne détaillons pas la construction de ce circuit car il est facile d'en donner un bien meilleur qui fait le même calcul. Pour le voir, nous introduisons deux nouvelles définitions : une évaluation est l'opération qui au polynôme $f(\underline{x}, u)$ associe le polynôme $f(\underline{x}, 0)$, ou bien le polynôme $f(\underline{x}, 1)$; une production est une opération du type $\Pi_{\underline{u} \in \{0,1\}^{\wedge n}} f(\underline{x}, \underline{u})$. Comme dans le cas des sommations, les productions se décomposent en productions unaires.

Evaluations, sommations et productions se simulent les unes les autres en présence des opérations arithmétiques, et de la constante -1. Cela est dû aux formules : $f(0) = \Sigma_u(1-u).f(u)$, $f(1) = \Sigma_u u.f(u)$; $f(0) = \Pi_u[(1-u).f(u)+u]$, $f(1) = \Pi_u[u.f(u)+1-u]$; $\Sigma_u f(u) = f(0) + f(1)$; $\Pi_u f(u) = f(0).f(1)$.

En conséquence, les suites de circuits de complexité polynomiale, comportant des portes binaires d'addition et de multiplication, et des portes unaires de sommation, ou des portes unaires de production, ou des portes unaires d'évaluation, ou les trois à la fois, définissent la même classe de suites de polynômes, celle que nous avons appelée VSPdl ! Nous avons choisi de la définir à partir de la sommation, opération naturelle si on considère qu'un polynôme est somme de ses monômes, qui en outre permet d'en distinguer la sous-classe VNPdl et d'énoncer notre Théorème 2. On peut penser toutefois, au vu de l'usage constant que nous avons fait des changements de variables booléennes, que les évaluations sont plus naturelles.

Comme la Factorielle s'exprime facilement par une production, on majore sa complexité avec une égale facilité :

THÉORÈME 5. *Factorielle$_n$ se calcule en* $13.n+2$ *opérations arithmétiques et une production, qu'on peut remplacer par* $2.n$ *sommations et* $4.n$ *opérations arithmétiques ; cela donne un circuit* S*-arithmétique de complexité* $17.n+2$*, comportant* $2.n$ *sommations de Valiant.*

DÉMONSTRATION. Factorielle$_n(\underline{u}) = \Pi_{\underline{v}}[f(\underline{u}, \underline{v}).(-1 + v_0 + 2.v_1 + \cdots + 2^{n-1}.v_{n-1}) + 1]$, où $f(\underline{u}, \underline{v})$ est un polynôme qui vaut 1 si l'entier représenté par $\underline{u}$ est supérieur ou égal à celui représenté par $\underline{v}$, et 0 sinon. On peut prendre pour f le terme f_n de la suite définie par la relation de récurrence suivante : $f_0 = 1$, $f_{i+1} = f_i.[1+(1-(u_{n-1}-v_{n-1})^2). \ldots .(1-(u_n-i-v_n-i)^2).(u_{n-i}-1-1).v_{n-i}-1]$, le polynôme f_i valant 1 si le nombre défini par les i chiffres de tête de $\underline{u}$ est supérieur ou égal à celui défini par les i chiffres de tête de $\underline{v}$, et 0 sinon. Il est alors calculable en $11.n+1$ opérations, d'où le premier résultat. Le second vient de ce qu'une production unaire se simule par deux sommations et quatre opérations arithmétiques. ⊣

Il est très peu probable, ou à tout le moins catastrophique pour la cryptographie à clef publique, que la Factorielle, et en conséquence la Pascaline, soient dans VPdl, car, comme il est bien connu, si la Factorielle se calcule en peu d'opérations on peut rapidement factoriser les nombres écrits en chiffres ; en effet, on détermine si N a un facteur premier inférieur à M en calculant le pgcd de N et du reste modulo N de M! ; si la Factorielle est dans VPdl, cela représente un calcul faisable en chiffres ; en quelques étapes on localise le plus petit facteur premier de N, et finalement on le factorise.

Cela milite en faveur de l'inégalité VSPdl $\neq$ VPdl. Par contre, la présence de la Pascaline ou de la Factorielle dans VNPdl ne semble pas avoir de conséquences aussi dramatiques, ce qui permet d'envisager avec moins d'effroi la possibilité de la clôture de la classe VNPdl pour la prise de fonction-coefficient, ou même l'égalité de VSPdl et de VNPdl.

Le seul fait de considérer des fonctions numériques conduit à se poser la question de savoir si une hypothèse du type VS(ou N)P = VP équivaut à sa restriction aux fonctions numériques (ou même aux suites numériques, qui sont les fonctions numériques dépendant de 0 variables). Plus précisément, on peut se demander si la Pascaline et la Factorielle sont universelles pour la classe VSPdl, relativement à une forme de réduction qui reste à définir.

Enfin, étant donné le rôle essentiel que jouent les soustractions dans nos calculs, il est peu probable qu'on puisse les éliminer ; mais il ne sera peut-être pas facile de répondre à la question suivante : est-ce que la Pascaline et la Factorielle sont calculables par des circuits S-arithmétiques de complexité polynomiale, qui n'utilisent que la constante 1 au lieu de la constante -1 ?

REMARQUE 3. Il n'y a qu'un nombre simplement exponentiel de circuits S-arithmétiques de complexité n, si bien qu'un simple argument de diagonalisation fabrique des suites d'entiers de croissance doublement exponentielle qui ne sont pas dans VSPdl ; plus précisément, on peut trouver une suite d'entiers positifs $a_n < 2 \wedge 2^n$ tels que, pour n assez grand, a_n ne soit pas calculable par un circuit S-arithmétique de complexité inférieure à $n^{\log(n)}$.

REMARQUE 4. Il est peu probable que nos calculs de la Factorielle présentent un quelconque intérêt pratique ; en effet, ils utilisent les sommations de Valiant pour obtenir plusieurs jeux de la même fonction, sans avoir à refaire le calcul : cela semble bien peu réaliste. Dans la sommation du changement de variable, à chaque $\underline{v}$ fixé il n'y a qu'une valeur de $\underline{u}$ pour laquelle la fonction que l'on somme sur $\underline{u}$ n'est pas nulle, si bien que toute tentative de calcul de cette sommation par échantillonage (la somme est le produit de la moyenne par 2^n, où n est le nombre de variables de sommation) aboutit à la fonction nulle. De même, le calcul du produit de convolution fait intervenir une sommation portant sur un ensemble négligeable. On n'oubliera pas que la problématique des classes de complexité n'est pas de construire des modèles de calcul réalistes, mais plutôt des modèles de calcul permettant de poser des questions significatives sur la réalité des calculs ; ce n'est pas tout-à-fait la même chose.

REMARQUE 5. La plupart des auteurs qui étudient le calcul de la Factorielle, ou d'autres fonctions numériques, les considèrent point par point. Ce serait une chose bien remarquable qu'il existe un polynôme p(n) tel que, pour tout entier N, le nombre N! se calcule à partir de -1 en p(log(N)) opérations arithmétiques ; mais on ne pourrait en déduire sans précautions que la Factorielle est dans VPdl. Ce qu'on pourrait en déduire, c'est que tout nombre de n chiffres, pris individuellement, se factorise en un temps polynomial en n, ce qui est absolument évident puisqu'il suffit de se donner les facteurs. Autrement dit, le problème de la factorisation ne peut se poser que si on introduit des variables booléennes.

§5. Les résultats de Malod en caractéristique p. Pour étudier les fonctions-coefficients de la classe VNPdl, Malod, dans [Malod, 2003], introduit un gigantesque polynôme VNPdl-universel, dont l'expression de la fonction-coefficient fait intervenir des boîtes noires calculant la Pascaline. Il observe que, si p est un nombre premier fixé, un théorème de Lucas datant de la fin du 19° siècle permet de calculer en temps polynomial le reste modulo p de la Pascaline (pour la Factorielle, c'est évident car il est nul si $N \geq p$).

Si donc on calcule non pas des polynômes à coefficients entiers, mais des polynômes à coefficients dans le corps $F_p = Z/pZ$, on voit que VNPdl(mod p) est close pour la prise de fonctions-coefficients.

En outre, Malod observe que la fonction-coefficient de son polynôme VNPdl(mod p)-universel est dans VNPt(mod p), si bien que toute suite de VNPdl(mod p) s'obtient à partir d'une suite de VNPmd(mod p) en remplaçant des variables par des puissances simplement exponentielles de variables. Il en déduit ce résultat remarquable : pour chaque nombre premier p fixé, l'hypothèse VNPdl(mod p) = VPdl(mod p) équivaut à l'hypothèse VNPmd(mod p) = VPmd(mod p).

Après avoir remarqué que toute hypothèse du type VP = VS(ou N) P implique, par un simple passage au quotient, sa contrepartie modulo p, nous ajoutons à ces résultats un modeste supplément.

LEMME 6. *En caractéristique p, toute fonction numérique qui est dans* VNPdl(mod p) *est dans* VNPmd(mod p).

DÉMONSTRATION. La démonstration s'inspire du lemme facile et bien connu de réduction parcimonieuse des circuits aux termes (NP = NPt, et #P = #Pt). Il suffit de montrer qu'une suite de fonctions numériques qui est dans VPdl est dans VNPmd. Soit $C(\underline{u})$ un circuit arithmétique à variables booléennes ; pour chaque $q \neq 0$ dans $F_p = Z/pZ$, on fabrique un terme arithmétique $T_q(\underline{u}, \underline{v})$, de complexité polynomiale en celle de C, tel que : si $C(\underline{u}) = q$, $T_q(\underline{u}, \underline{v}) = q$ pour une seule valeur de $\underline{v}$, et 0 pour les autres ; si $C(\underline{u}) \neq q$, $T_q(\underline{u}, \underline{v}) = 0$ pour tout $\underline{v}$. Le uplet $\underline{v}$ représente les valeurs qui sont calculées aux portes de C, si bien que la construction de ce terme demandera un codage binaire des éléments de F_p, par exemple par des $(p-1)$-uplets booléens du type $(1, \ldots, 1, 0, \ldots, 0)$. Le circuit $C(\underline{u})$ calcule la même fonction que $\Sigma_{\underline{v}} T_1(\underline{u}, \underline{v}) + \cdots + T_{p-1}(\underline{u}, \underline{v})$. ⊣

Une fois qu'on sait que la classe VNPdl(mod p) est close pour la prise de fonctions-coefficients, les autres résultats de Malod suivent aisément de ce lemme. En effet, si f_n est une suite dans VNPdl(mod p), la suite de ses polynômes réduits est, d'après le lemme, dans VNPmd ; comme f_n s'obtient par substitution de puissances simplement exponentielles de variables dans son polynôme réduit, on voit que VNPdl(mod p) = VPdl(mod p) équivaut à VNPmd(mod p) = VPmd(mod p). On obtient également ceci :

THÉORÈME 7. *En caractéristique p,* VSPdl(mod p) = VPdl(mod p) *si et seulement si cette hypothèse est vérifiée par les fonctions numériques et en outre* VNPmd(mod p) = VPmd(mod p).

DÉMONSTRATION. D'un côté, VSPdl = VPdl implique VNPdl = VPdl et VNPdb = VPdb ; mais, en caractéristique p, puisqu'il n'y a qu'un nombre fini

d'entiers modulo p, VPdb = VPmd, si bien qu'on obtient VNPmd = VPmd. Réciproquement, la première hypothèse implique que la fonction-coefficient d'une suite de VSPdl est dans VPdl, donc dans VNPmd, et son polynôme réduit aussi. La deuxième hypothèse met ce dernier dans VPmd. ⊣

Nous allons voir que l'hypothèse VSPdl(mod p) = VPdl(mod p) équivaut à celle du problème algorithmique majeur du début de notre siècle : elle est donc très improbable.

La clôture de la classe VNPdl(mod p) pour la prise de fonctions-coefficients peut inciter à considérer favorablement l'hypothèse VSPdl(mod p) = VNPdl(mod p) ; elle semble toutefois presque aussi peu probable que sa version de caractéristique nulle.

§6. Koiran et l'effondrement hypothétique du calcul de la Factorielle. Nous revenons à la caractéristique nulle.

Pascal Koiran a été le premier, semble-t'il, à évaluer l'incidence, sur la complexité de calcul de certaines suites numériques, de combinaisons d'hypothèses portant sur des classes de complexité à la Valiant et d'hypothèses portant sur des classes de complexité classiques, c'est-à-dire booléennes. C'est ainsi que, dans [Koiran, 2004], il montre entre autres choses que, si VNPmd = VPmd et P = PSPACE, alors la Factorielle est calculable en un nombre polynomial d'opérations arithmétiques, c'est-à-dire qu'elle est dans VPdl.

A strictement parler, Koiran montre un résultat un peu plus faible, puisqu'il calcule les nombres N! individuellement. Ce n'est qu'un point de détail, car il est facile de voir que ses arguments sont valides uniformément pour tous les nombres de n chiffres ; et de toutes façons, nous allons montrer ici que ses hypothèses n'entraînent pas seulement l'effondrement du calcul de quelques suites, mais bien l'effondrement de la classe VSPdl toute entière.

La classe P (non uniforme !) est formée des suites de fonctions booléennes calculables en un nombre polynomial d'opérations ; comme cette définition ne dépend pas du choix des opérations de base, il s'agit de la classe VPdl(mod 2) restreinte aux fonctions numériques.

Nous voyons aussi apparaître la classe PSPACE des suites de fonctions booléennes calculables en espace polynomial. Sous sa version non-uniforme, elle est constituée des suites de fonctions booléennes calculées par une suite de circuits de complexité polynomiale, comprenant des portes d'addition et de multiplication modulo deux, ainsi que des portes unaires de quantifications existentielles : $(\exists u) f(\underline{x}, u)$ vaut 1 s'il existe un u (booléen) tel que $f(\underline{x}, u) = 1$, et 0 sinon ; cette opération se simule dans $\mathbb{Z}/2\mathbb{Z}$ par $1 + (1 + f(0)).(1 + f(1))$, tandis que $f(0) = (\exists u)(1 + u).f(\underline{x}, u)$ et $f(1) = (\exists u)u.f(\underline{x}, u)$. A la place de quantifications existentielles, on peut aussi bien utiliser des évaluations, ou des sommations, ou des productions (ces dernières représentent des quanteurs universels). Autrement dit la classe PSPACE est la classe VSPdl(mod 2) restreinte aux fonctions numériques.

Parmi les mille et une façons de définir PSPACE, nous avons choisi celle qui nous convient le mieux ; elle est justifiée par la PSPACE-complétude de l'évaluation des formules avec quantifications.

La définition de PSPACE à partir des circuits comportant des portes d'addition et de multiplication modulo 2, ainsi que des portes d'évaluation, est apparue dans [Babai and Fortnow, 1991].

La philosophie de tout ça, c'est que la classe P est décrite par des opérations, tandis la classe PSPACE l'est par des opérations et des quantifications (c'est à dire des opérateurs dépendant de l'ensemble des valeurs que les fonctions prennent lorsque varie le uplet de variables quantifiées). Les quantifications existentielles et universelles sont propres au calcul booléen ; on fabrique à partir d'elles des énoncés directement adaptés à la description d'une foule de choses, dont le fonctionnement des algorithmes ; leur remontée en tête des circuits permet de définir la hiérarchie polynomiale : c'est une propriété qui ressemble à celle des sommations dans le calcul des polynômes. On peut leur préférer les évaluations, qui s'appliquent à un contexte plus général, et qui sont si typiques de PSPACE : en effet, il ne faut pas plus d'espace pour calculer $f(0)$ ou $f(1)$ que pour calculer $f(u)$.

On ne sait pas séparer P de PSPACE ; la très improbable égalité P = PSPACE équivaut, par simple changement de sigle, à VSPdl(mod 2) = VPdl(mod 2) restreinte aux fonctions numériques, soit encore booléennes.

Nous pouvons évaluer la complexité d'une fonction booléenne aussi bien au sens classique, booléen, qu'au sens du calcul des polynômes, à la manière de Valiant, en caractéristique nulle (0 et 1 étant alors considérés comme des entiers, et non pas comme des entiers modulo 2). Voici comment ces deux calculs de complexité se raccordent :

THÉORÈME 8. *Les fonctions booléennes qui sont dans* VPdl *sont exactement celles qui sont dans* P (*version non uniforme*). *Les fonctions booléennes qui sont dans* VSPdl *sont exactement celles qui sont dans* PSPACE (*idem*).

DÉMONSTRATION. Si une fonction booléenne est donnée par un circuit arithmétique, comme on sait que le résultat vaut 0 ou 1, il suffit de faire le calcul modulo 2 ; réciproquement, pour simuler des additions et des multiplications modulo 2 par des opérations portant sur les entiers, restreintes à $\{0, 1\}$, il suffit de remplacer les additions par $x + y - 2.x.y$.

Si une fonction booléenne est donnée par un S-circuit, son calcul modulo 2 donne un S-circuit modulo 2, caractérisant la classe PSPACE. Pour la réciproque, on définit la classe PSPACE par des circuits comportant des sommes, des produits et des évaluations modulo deux ; pour les simuler par des opérations portant sur les entiers, restreintes à $\{0, 1\}$, il suffit de remplacer les additions par $x + y - 2.x.y$. ⊣

REMARQUE 6. La même démonstration vaut pour la classe VPt, associée à la profondeur logarithmique : la partie booléenne de (c'est-à-dire l'ensemble des fonctions booléennes de) VPt(mod 2) comme celle de VPt est la classe Pt des suites de fonctions booléennes calculable en profondeur logarithmique. Par contre, comme la simulation de la somme modulo 2 fait intervenir une multiplication, il est possible que la partie booléenne de VPmd soit un sous-ensemble propre de celle de VPmd(mod 2). Par ailleurs, comme on ne peut pas simuler directement une sommation modulo 2 par une sommation en caractéristique nulle, on ne voit pas de raison pour que VNPmd et VNPmd(mod 2) d'une part, VNPdl et VNPdl(mod 2) d'autre part, aient les mêmes parties booléennes.

REMARQUE 7. Si p est un nombre premier quelconque, PSPACE est la partie booléenne de VSPdl(mod p), P la partie booléenne de VPdl(mod p), et Pt la partie booléenne de VPt(mod p) ; la démonstration n'est pas aussi directe, car si $p \geq 3$ on ne peut simuler la somme, ni d'ailleurs le produit, modulo p par des polynômes à coefficients entiers, mais on peut simuler les calculs dans $\mathbf{Z}/p\mathbf{Z}$ en représentant les éléments de ce corps par des uplets booléens. Cela vient de ce que les classes PSPACE, P et Pt ont des définitions solides, indépendantes de la base d'opérations choisie. Par contre, les autres sont spécifiquement liées à la somme et au produit, si bien qu'il est possible que les parties booléennes de VPmd(mod p), VNPmd(mod p), VNPdl(mod p) varient avec p.

Pour ramener la complexité des polynômes à celle des fonctions booléennes, nous définissons maintenant la fonction-chiffre.

Si $f(\underline{u})$ est une fonction-numérique positive, sa fonction-chiffre $\mathrm{chf}(\underline{u}, \underline{v})$ donne le développement en base deux de l'entier $f(\underline{u})$: $\mathrm{chf}(\underline{u}, \underline{v})$ est le N° chiffre, valant 0 ou 1, de $f(\underline{u})$, où N est l'entier dont $\underline{v}$ est l'écriture en base deux. Autrement dit $f(\underline{u}) = \Sigma_{\underline{v}} \mathrm{chf}(\underline{u}, \underline{v}).2^{\underline{v}} = \Sigma_{\underline{v}} \mathrm{chf}(\underline{u}, \underline{v}).2^{v0}.2^{2.v1}. \ldots .2 \wedge (2^{n-1}.v_{n-1})$.

Comme nous ne considérons que des suites d'entiers au pire doublement exponentielles, leurs fonctions-chiffres ne dépendent que d'un nombre polynomial d'arguments.

Nous appelons polynôme-chiffre de $f(\underline{u})$ le polynôme $\mathrm{pchf}(\underline{u}, \underline{z}) = \Sigma_{\underline{v}} \mathrm{chf}(\underline{u}, \underline{v}).z_0{}^{v0} \ldots .z_{n-1}{}^{vn-1}$. Comme $z^v = v.(z-1)+1$, ce polynôme est du premier degré en chacune de ses inconnues non-booléennes, et $f(\underline{u})$ s'obtient en y remplaçant z_i par $2 \wedge 2^i$.

La fonction-chiffre d'une fonction numérique quelconque $f(\underline{u})$ est constituée de la donnée de la fonction-chiffre de sa valeur absolue, et de sa fonction-signe $\mathrm{sgf}(\underline{u})$, qui vaut 1 si $f(\underline{u}) \geq 0$, et 0 sinon. La fonction-chiffre d'un polynôme à coefficients entiers relatifs est celle sa fonction-coefficient.

Nous attachons ainsi, de manière bijective, un objet booléen à un polynôme, mais le passage du polynôme à sa fonction-chiffre, ainsi que le passage inverse, provoquent des augmentations de complexité qu'il nous faut controler, et qui justifient l'introduction de la classe VSPdl.

Le polynôme-chiffre d'un polynôme à coefficients dans Z s'obtient à partir de son polynôme réduit par remplacement de sa fonction-coefficient par le polynôme-chiffre de cette dernière ; ses coefficients valent tous 0, 1 ou -1, et il est de degré un par rapport à chacune de ses inconnues. Un polynôme s'obtient par substitution de puissances de deux et de variables dans son polynôme-chiffre. Quand on a affaire à une fonction-polynôme, on obtient ainsi une fonction-polynôme-chiffre !

Nous sommes maintenant armés pour montrer ce renforcement du Théorème 3 :

THÉORÈME 9. *La classe* VSPdl *est stable pour la prise de fonctions-chiffres, soit encore : la suite des fonctions-chiffres d'une suite de polynômes de* VSPdl *est dans* PSPACE(/*poly*).

DÉMONSTRATION. Il nous faut reprendre la démonstration du Théorème 3, en remplaçant le calcul en nombre des coefficients par leur calcul en chiffres (on ne peut pas manipuler simultanément tous ces chiffres, car il y en a une quantité exponentielle !). Comme il faut gérer un problème pénible de retenues, nous serons moins précis que lors des démonstrations des théorèmes précédents, en nous contentant

d'indiquer pourquoi les calculs que nous décrivons se font en espace polynomial : il nous suffit de convaincre notre lecteur qu'il peut les faire sur un petit tableau noir, à condition de disposer d'un gros stock de craie et de ne pas rechigner à manipuler souvent le chiffon ; nous lui laissons le soin, s'il le désire, de représenter ce calcul par un S-circuit dont il évaluera la complexité.

Dans un premier temps, nous considérons un S-circuit positif C, c'est-à-dire qui utilise la constante $1 = (-1)^2$ au lieu de -1 à ses entrées, dont nous notons c la complexité. Pour l'expression des fonctions-chiffres, nous considérons le polynôme calculé en p comme un polynôme en toutes les variables de C, et pas seulement en celles qui sont libres en p.

Il est facile de calculer la fonction-chiffre d'une entrée, si bien qu'il ne nous reste plus qu'à montrer que la fonction-chiffre d'une porte d'opération se calcule en espace polynomial en c à partir de celle de la porte, ou de celles des deux portes, dont elle reçoit ses flèches.

Si p est une porte d'addition, $p = p' + p''$, il faut additionner en chiffres la fonction-chiffre de p' et celle de p'' (autrement dit, à partir de deux fonctions booléennes $f(\underline{u})$ et $g(\underline{u})$, il nous faut calculer en espace polynomial la fonction-chiffre de la somme des deux nombres dont f et g sont les fonctions-chiffres) ; bien que ces fonctions représentent des (uplets de) nombre à 2^c chiffres, ça se fait en espace polynomial car, si on additionne les nombres par la méthode de l'école primaire, pour calculer le $\underline{u}^\circ$ chiffre d'une somme on n'a pas besoin de garder en mémoire tout le calcul qui a été fait auparavant, mais seulement la retenue qui arrive de la droite (cette addition se représente facilement par un circuit avec quantifications, car il y a une retenue en $\underline{u}^\circ$ position si et seulement s'il existe une position antérieure $\underline{v}$ où les deux chiffres valent 1, telle que pour chaque position strictement intermédiaire les chiffres ne soient jamais tous les deux 0).

Si p est une porte de sommation, nous pouvons, quitte à la décomposer, supposer qu'elle ne porte que sur une variable, $p = \Sigma_u p'$; nous commençons par réintroduire les puissances de u en multipliant la fonction-chiffre de p' par $u^{\underline{v}}$, où $\underline{v}$ est le uplet de variables représentant le degré de u ; ensuite, nous effectuons une sommation en chiffre sur $(u, \underline{v})$: nous introduisons $c + 1$ portes pour calculer le résultat ; comme la sommation en chiffres est associative, nous pouvons évaluer les variables de $(u, \underline{v})$ une par une en 0 et en 1, ajouter un petit circuit simulant l'addition de deux nombres de $c + 1$ chiffres, et passer à la variable suivante. Nous obtenons ainsi une sorte de fonction-chiffre relativement à toutes les variables sauf u, si ce n'est que ses pseudo-chiffres sont compris entre 0 et $2^c + 1$, étant chacun exprimé par $c + 1$ chiffres. Comme il y a $c + 1$ portes pour représenter ces chiffres $(y_0, \ldots, y_n)$, nous obtenons en fait $c + 1$ fonctions $ch_0(\underline{v}) = y_0, \ldots, ch_n(\underline{v}) = y_n$; nous effectuons alors des changements de variables pour décaler celles de $ch_i(\underline{v})$ de i pas, afin que que y_i soit à la bonne place. Nous obtenons ainsi $ch'_0(\underline{v}), \ldots, ch'_n(\underline{v})$, que nous additionnons en chiffres : cette fois, les pseudo-chiffres sont compris entre 0 et $c + 1$, si bien qu'à l'étape suivante on aura au plus $\log(c + 1) + 1$ fonctions. Comme $2^m > m$ si $m > 2$, on obtient rapidement des chiffres compris entre 0 et 2, que nous traitons comme dans le cas où p est une porte d'addition : on décale les retenues et on additionne les deux fonctions-chiffres. Il ne nous reste plus qu'à réintroduire les variables représentant le degré de la variable u.

Si p est une porte de multiplication, $p = p'.p''$, il nous faut effectuer le produit de convolution des fonctions-chiffres de p' et de p''. On commence par changer les variables. On est ensuite confronté au problème suivant : étant données deux fonctions booléennes $f(\underline{u})$ et $g(\underline{v})$, il faut calculer en espace polynomial la fonction-chiffre du produit des nombres dont f et g sont les fonctions-chiffres. Pour cela, il faut utiliser un algorithme parallélisé de multiplication. Multiplier ces deux nombres par la méthode de l'école primaire revient à faire un nombre exponentiel d'additions ; on multiplie $f(\underline{u})$ par $g(\underline{v})$, et on décale ces chiffres de $\underline{v}$ places, ce qui donne une fonction $h(\underline{u}, \underline{v})$: il nous faut calculer la fonction-chiffre de la somme des nombres dont la fonction-chiffre est $h(\underline{u}, \underline{v})$, pour un $\underline{v}$ fixé. Pour paralléliser le processus, nous faisons des additions d'Ofman (voir [Karatsuba and Ofman, 1963]) qui remplacent, en profondeur constante, quatre nombres donnés en chiffres par deux nombres de même somme ; pour cela, on additionne chiffre par chiffre les trois premiers modulo deux, puis on calcule un deuxième nombre formé des retenues ; on fait ensuite la même chose avec ces deux nombres et le quatrième (on remarque que les $\underline{u}^\circ$ chiffres du résultat ne dépendent que des chiffres de rang $\underline{u}$, $\underline{u} - 1$ et $\underline{u} - 2$ des quatre nombres de départ, ce qui facilite la représentation de cette opération par un petit circuit avec changements de variables) ; en effectuant l'addition d'Ofman de $h(\underline{u}, \underline{v}', 0, 0)$, $h(\underline{u}, \underline{v}', 0, 1)$, $h(\underline{u}, \underline{v}', 1, 0)$ et de $h(\underline{u}, \underline{v}', 1, 1)$, on remplace $h(\underline{u}, \underline{v})$ par une fonction $h_1(\underline{u}, \underline{v}_1)$, où la longueur de $\underline{v}_1$ est inférieure d'une unité à celle de $\underline{v}$; après avoir répété l'opération $c + 1$ fois, il nous reste deux nombres, que nous additionnons. Quant à la sommation exprimant le produit de convolution, elle est traitée comme ci-dessus.

Si le circuit n'est pas positif, on lui associe un circuit positif trois fois plus grand, contenant, pour chaque porte p de C, une porte p^+ et une porte p^- telles que le nombre caculé en p soit la différence du polynôme calculé en p^+ et du nombre calculé en p^- (aux portes d'addition et de sommation on somme les parties positives et les parties négatives ; si $p = p'.p''$, $p^+ = p'^+.p''^+ + p'^-.p''^-$, $p^- = p'^-.p''^+ + p'^+.p''^-$). Ayant obtenu les fonctions-chiffres des coefficients de la partie positive et de la partie négative de la sortie de C, on en déduit sa fonction-signe en espace polynomial, en examinant les chiffres un par un à partir des plus hauts ; on obtient ensuite la fonction-chiffre des valeurs absolues de ses coefficients par une sommation. ⊣

D'après ce théorème, VSPdl est formée des suites de polynômes à coefficients entiers, de nombre de variables, degré et logarithme d'ampleur polynomiaux, et dont la fonction-chiffre est dans PSPACE ; cette classe a été introduite par Koiran et Perifel dans [Koiran and Perifel, 2007], auquel on préférera la remarquable rédaction en français de [Perifel, 2007], sous le nom de VPSPACE (ou plus exactement VPSPACE$^\circ$/poly). Comme le montre la suite, son étude est grandement facilitée par sa définition en termes de S-circuits.

REMARQUE 8. Le passage par la fonction-chiffre montre que l'adjonction aux circuits de portes de changement de variables non-booléennes ne fait pas sortir de la classe VSPdl. Remarquons aussi que des évaluations en $a_1, \ldots, a_n$ se ramènent à des évaluations booléennes par des changements de variables $u_0.x + u_1.a_1 + \cdots + u_n.a_n$. Les changements de variables devraient jouer un rôle décisif dans toute tentative de définition de la complexité d'espace pour les calculs dans une structure arbitraire introduits par [Goode, 1994] et [Poizat, 1995].

Voici maintenant le pendant du Lemme 6 :

LEMME 10. *La classe* P *est incluse dans* VNPmd.

DÉMONSTRATION. La classe P est caractérisée par des circuits de complexité polynomiale comportant des portes d'addition et de multiplication modulo deux, qu'on simule respectivement par $x + y - 2.xy$ et $x.y$. Pour controler le degré de la simulation, nous utilisons la réduction parcimonieuse des circuits aux termes. Elle associe au circuit booléen $C(\underline{u})$ un terme booléen $T(\underline{u}, \underline{v})$ de complexité un peu plus grande tel que $T(\underline{u}, \underline{v}) = 0$ pour tous les $\underline{v}$ sauf un si $C(\underline{u}) = 1$, tandis que $T(\underline{u}, \underline{v}) = 0$ pour tous les $\underline{v}$ si $C(\underline{u}) = 0$; autrement dit $C(\underline{u}) = \Sigma_{\underline{v}} T(\underline{u}, \underline{v})$; cette sommation modulo deux se simule par une sommation en caractéristique 0, puisqu'elle porte sur des valeurs toutes nulles sauf au plus une, qui vaut 1. Si on parallélise les suites de termes, leur simulation sera dans VPmd, et celle des suites de circuits correspondants dans VNPmd. ⊣

COROLLAIRE 11. *Si* P = PSPACE, *alors* VSPdl = VNPdl ; *plus précisément, sous cette improbable hypothèse, toute suite de* VSPdl *s'obtient à partir d'une suite de* VNPmd *par substitution de puissances simplement exponentielles de variables et de* 2.

DÉMONSTRATION. D'après le Lemme 10, si P = PSPACE, le polynôme-chiffre d'une suite de VSPdl est dans VNPmd. ⊣

Et voici la généralisation de l'effondrement koiranien du calcul de la factorielle :

COROLLAIRE 12. *Si* P = PSPACE *et* VNPmd = VPmd (*ou* VNPdl = VPdl), *alors* VSPdl = VPdl.

DÉMONSTRATION. Sous ces hypothèses, le polynôme-chiffre d'une suite de VSPdl est dans VPdl. ⊣

L'hypothèse P = PSPACE est bien sûr très improbable, mais le scandale algorithmique du millénaire est qu'on ne sait pas la réfuter. La signification de ces corollaires, c'est que, si vous croyez, comme il est légitime, que VNPdl est une sous-classe propre de VSPdl, ou bien que la factorielle n'est pas calculable, vous aurez beaucoup de mal à faire passer cette croyance du domaine de la Foi à celui de la Raison.

Réciproquement, il est clair que VSPdl = VPdl implique P = PSPACE et VNPdl = VPdl ; mais il n'est pas clair que cette dernière égalité implique VNPmd = VPmd, car le calcul en degré borné d'une suite de polynômes de sous-degré borné peut demander l'emploi de grosses constantes entières (un problème qui ne se pose pas en caractéristique p).

On peut aussi se demander s'il est possible de montrer que VNPmd = VPmd implique VNPdl = VPdl (on observera que c'est vrai sous l'hypothèse P = SPACE !). Voici un autre théorème, dont la protase est aussi improbable que l'apodose, qui va dans ce sens :

LEMME 13. *Si* VNPmd = VPmd, *toute suite de fonctions booléennes qui est dans* VNPdl *est dans* VPmd.

DÉMONSTRATION. Le calcul d'une fonction booléenne peut se faire modulo deux, ce qui remplace la sommation par un calcul de parité. En réduisant parcimonieusement les circuits booléens aux termes booléens, on obtient, à partir d'une suite f_n

de fonctions booléennes dans VNPdl, une suite g_n de fonctions numériques dans VNPmd, telle que la valeur de f_n soit le reste modulo deux de l'entier calculé par g_n. Par hypothèse, g_n est dans VPmd ; si dans un circuit arithmétique calculant g_n on remplace les portes d'additions par des additions modulo 2 de la forme $x+y-2.xy$, on calcule f_n. En conséquence f_n est dans P, soit encore, d'après le Lemme 10, dans VNPmd = VPmd. ⊣

Ce dernier lemme montre que, sous l'hypothèse VNPmd = VPmd, la classe P est incluse dans VPmd, classe que [Valiant, Skyumi, Berkowitz, and Rackoff, 1983] parallélise en profondeur $O(\log^2 n)$ tout en préservant sa complexité polynomiale ; si on y ajoute P = PSPACE, on aboutit à un effondrement sévère de PSPACE, qui, d'après [Koiran and Perifel, 2007], n'a pas été réfuté, même pour des classes d'uniformité polynomiale.

L'hypothèse VSPdl = VPdl provoque elle-aussi cet effondrement ; en effet, elle implique sa contrepartie modulo 2, qui implique que VNPmd(mod 2) = VPmd(mod 2), qui implique à son tour que P est la partie booléenne de VPmd(mod 2).

LEMME 14. *Si* VNPmd = VPmd, *toute suite de polynômes de* VPdl, *de degré polynomial et dont les coefficients sont des entiers simplement exponentiels, est dans* VPmd.

DÉMONSTRATION. Grâce au calcul des parties homogènes, la suite f_n se calcule par des petits circuits de petit sous-degré ; elle est donc calculable par des petits circuits multiplicativement disjoints $C_n(\underline{x}_n, \underline{a}_n)$ utilisant des constantes entières $\underline{a}_n$ doublement exponentielles. La fonction-coefficient partielle de $C_n(\underline{x}_n, \underline{a}_n)$ est dans VNPmd = VPmd ; si les coefficients de f_n sont compris entre -2^m et 2^m, on peut remplacer les a_n par leur reste modulo 2^{2m}, et faire les calculs en chiffres pour qu'ils restent dans cet intervalle, ce qui représente un calcul dans VPdl, même si on emploie des algorithmes naïfs pour l'addition et la multiplication en chiffres. Comme il s'agit d'une fonction booléenne, la fonction-chiffre de la suite f_n est dans VNPt, si bien que f_n est dans VNPmd = VPmd. ⊣

Comme aucune méthode connue ne permet de tronquer les coefficients d'un polynôme à la manière de celle utilisée pour se débarasser de ses monômes de haut degré, ni simuler un calcul de polynômes modulo p par un calcul en caractéristique nulle, la conclusion du Lemme 14 semble improbable ; ce lemme indique qu'elle sera dure à réfuter.

LEMME 15. *Si* VSPdl = VPdl *et* VNPmd = VPmd, *toute suite de polynômes de complexité polynomiale, à coefficients entiers simplement exponentiels, est calculable par une suite de circuits de complexité polynomiale et d'ampleur simplement exponentielle.*

DÉMONSTRATION. La première hypothèse met la fonction-chiffre dans VPdl, et la deuxième met le polynôme réduit dans VPmd. ⊣

Terminons par la mention d'un résultat tout récent : Peter Bürgisser, dans [Bürgisser, 2007], améliore considérablement [Koiran, 2004] en plaçant la Factorielle dans une hiérarchie qui s'effondre sous la seule hypothèse VNPmd = VPmd (on remarquera que ni Koiran, ni Bürgisser ne placent sans hypothèses la Factorielle dans

VNPdl). Il ne semble pas déraisonnable d'espérer que la conjonction des méthodes de Malod et de Bürgisser, ainsi que certains résultats de [Perifel, 2007], permettent d'éclairer un jour les rapports entre les classes VNPmd et VNPdl.

RÉFÉRENCES

[1991] LAZLO BABAI and LANCE FORTNOW, *Arithmetization : a new method in structural complexity theory*, ***Computational Complexity***, vol. 1 (1991), pp. 41–66.

[2000] PETER BÜRGISSER, ***Completeness and reduction in algebraic complexity theory***, Springer Verlag, 2000.

[2007] ———, ***On defining integers in the counting hierarchy and proving lower bounds in algebraic complexity***, Lecture Notes on Computer Science, vol. 4393, 2007.

[1994] JOHN B. GOODE, *Accessible telephone directories*, this JOURNAL, vol. 59 (1994), pp. 92–105.

[1963] KARATSUBA and OFMAN, *Multiplication sur des automates de nombres à plusieurs chiffres*, ***Soviet Physics – Doklady***, vol. 7 (1963), pp. 595–596, en russe.

[2004] PASCAL KOIRAN, *Valiant's model and the cost of computing integers*, ***Computational Complexity***, vol. 13 (2004), pp. 131–146.

[2007] PASCAL KOIRAN and SYLVAIN PERIFEL, **VPSPACE** ***and a transfer theorem over the Reals***, Lecture Notes on Computer Science, vol. 4393, 2007.

[2003] GUILLAUME MALOD, ***Polynômes et coefficients***, thèse de doctorat, Université Claude Bernard, 2003.

[2006] GUILLAUME MALOD and NATACHA PORTIER, ***Characterizing Valiant's algebraic complexity classes***, Lecture Notes in Computer Sciences, vol. 4162, 2006, 267–279 ; version étendue à paraître au ***Journal of Complexity***.

[1976] DAVID E. MULLER and FRANCO P. PREPARATA, *Restructuring of arithmetic expressions for parallel evaluation*, ***Journal of the Association for Computing Machinery***, vol. 23 (1976), pp. 534–543.

[2007] SYLVAIN PERIFEL, ***Problèmes de décision et d'évaluation en complexité algébrique***, thèse de doctorat, Université de Lyon, 2007.

[1995] BRUNO POIZAT, ***Les petits cailloux***, Aléas, 1995.

[1979] LESLIE G. VALIANT, *Completeness classes in algebra*, ***Proceedings of the 11th ACM STOC***, ACM, 1979, pp. 249–261.

[1982] ———, *Reductibility by algebraic projections*, ***L'Enseignement Mathématique***, vol. 28 (1982), pp. 253–268, vol. 30, pp. 365–380.

[1983] L. G. VALIANT, S. SKYUMI, S. BERKOWITZ, and C. RACKOFF, *Fast parallel computations using few processors*, ***SIAM Journal for Computations***, vol. 12 (1983), pp. 641–644.

INSTITUT CAMILLE JORDAN
UNIVERSITÉ CLAUDE BERNARD
43, BOULEVARD DU 11 NOVEMBRE 1918
69622 VILLEURBANNE CEDEX, FRANCE
E-mail: poizat@math.univ-lyon1.fr