Polynômes donnés par des circuits algébriques et généralisation du modèle de Valiant

Sylvain Perifel
Stage de DEA avec Pascal Koiran, ENS Lyon

Le 30 juin 2004

Table des matières

Introduction

Les circuits booléens offrent une vision nouvelle de la théorie de la complexité, complétant celle des machines de Turing. Dans le cadre de la complexité algébrique, on définit de manière analogue des circuits algébriques grâce à des portes d'addition et de multiplication. On travaille alors sur un corps K fixé. Les entrées sont des variables x_i ou des constantes, et on calcule ainsi des polynômes $f \in K[x_1, \ldots, x_n]$ à plusieurs indéterminées à coefficients dans K.

Plus formellement, un *circuit algébrique* est un circuit orienté sans cycle, dont les sommets sont de degré entrant 0 ou 2, et dont un et un seul sommet est de degré sortant nul. Ce dernier est appelé *sortie* ; les sommets de degré entrant nul sont appelés *entrées*. Les sommets qui ne sont pas des entrées sont appelés *portes*, et sont étiquetés par $+$ ou $\times$. Les entrées sont étiquetées par des variables ou des constantes de K. Dans ce qui suit, nous nous autoriserons seulement les constantes -1, 0 et 1.

Le polynôme calculé par la porte d'un circuit est défini de manière naturelle par induction :

- le polynôme calculé par une entrée est son étiquette (une constante ou une variable) ;
- le polynôme calculé par une porte $+$ recevant une flèche des portes α et β est la somme des polynômes calculés par α et β ;
- de même, le polynôme calculé par une porte $\times$ recevant une flèche des portes α et β est le produit des polynômes calculés par α et β.

Le *polynôme calculé par un circuit* est le polynôme calculé par sa sortie. La taille d'un circuit algébrique C est notée $|C|$ et représente le nombre de sommets de C.

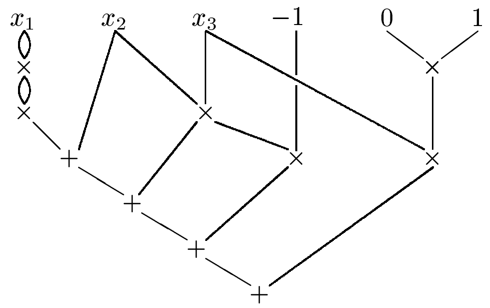

FIG. 1 – Un exemple de circuit de taille 16 calculant (de manière compliquée) $x_1^4 + x_2$.

Puisque nous n'autorisons pas d'autres constantes que -1, 0 et 1, les coefficients de nos polynômes resteront dans l'anneau engendré par ces constantes, c'est-à-dire $\mathbf{Z}$ dans le cas d'un corps de caractéristique nulle, ou le corps premier $\mathbf{Z}/p\mathbf{Z}$ dans le cas d'un corps de caractéristique $p > 0$. Ainsi, nos résultats dépendront souvent de la caractéristique du corps dans lequel on travaille.

Dans ce rapport, nous nous intéressons à deux grandes problématiques sur les circuits algébriques. Dans le premier chapitre, nous les abordons du point de vue de la complexité algébrique : nous définissons une hiérarchie de classes de complexité, décrivons leurs propriétés de base, et étudions les relations entre les classes. Cette hiérarchie s'inspire des classes VP et VNP de Valiant que nous définissons maintenant. Il s'agit de classes de suites de polynômes.

On dit qu'une suite de polynômes $(f_n)_{n\in\mathbf{N}}$ est *calculée par* la suite de circuits $(C_n)_{n\in\mathbf{N}}$ si pour tout $n \geq 0$, f_n est calculé par C_n. Pour simplifier les notations, on supprimera la référence $n \in \mathbf{N}$

lorsqu'il n'y a pas d'ambiguïté : ainsi, on écrira (f_n) pour la suite $(f_n)_{n\in\mathbf{N}}$. Selon la tradition informatique, on notera $p(n)$ pour désigner un polynôme en la variable n, plutôt que p seulement. De même, pour les polynômes en plusieurs variables, pour plus de clarté on pourra faire apparaître les variables : $(f_n(x_1,\ldots,x_{p(n)}))$ plutôt que (f_n).

La classe VP ([21]) est l'ensemble des suites de polynômes de degré polynomial calculées par une suite de circuits de taille polynomiale : c'est-à-dire que $(f_n)\in$ VP si et seulement s'il existe un polynôme $p(n)$ et une suite (C_n) de circuits calculant (f_n), et vérifiant pour tout $n\geq 0$:

- $|C_n|\leq p(n)$;
- le degré de f_n est majoré par $p(n)$.

Par l'hypothèse sur la taille des circuits, le nombre de variables de f_n est majoré par $p(n)$. Intuitivement, VP capture l'ensemble des suites de polynômes «facilement calculables». Valiant autorise en entrée des constantes arbitraires du corps sur lequel on travaille, alors que nous n'autoriserons que -1, 0 et 1.

L'autre classe de Valiant est VNP. Intuitivement, elle capture l'ensemble des suites de polynômes «facilement définissables». On ne sait pas effectuer des sommes sur un nombre exponentiel de termes dans VP, d'où l'idée de permettre ce genre de grosses sommations. Ainsi, par définition une suite $(f_n(x_1,\ldots,x_{u(n)}))$ est dans VNP s'il existe un polynôme $p(n)$ et une suite $(g_n(x_1,\ldots,x_{u(n)},y_1,\ldots,y_{p(n)}))\in$ VP tels que pour tout $n\geq 0$,

$$f_n(x_1,\ldots,x_{u(n)})=\sum_{\overline{\epsilon}\in\{0,1\}^{p(n)}} g_n(x_1,\ldots,x_{u(n)},\epsilon_1,\ldots,\epsilon_{p(n)}).$$

Bien entendu, la question ouverte principale est de savoir si VP = VNP. On pourra consulter [6] ou [15] pour approfondir la théorie de Valiant. L'article original de Valiant est [21].

Dans notre étude, rappelons-le, nous n'autorisons pas les constantes arbitraires, mais seulement -1, 0 et 1. De plus, nous ne nous restreignons pas aux suites de polynômes de degré polynomialement borné. Guillaume Malod, dans sa thèse [15], a déjà étudié les classes de Valiant dans ce contexte : il s'agit des classes $\mathrm{VP}^0_{\mathrm{nb}}$ et $\mathrm{VNP}^0_{\mathrm{nb}}$, définies de la même façon que précédemment. L'indice signifie «non borné» (pour le degré), et l'exposant 0 signifie «sans constante».

En s'appuyant sur la remarque que dans ces classes, nous ne savons toujours pas calculer les gros produits (par exemple, la factorielle), nous introduisons la hiérarchie des classes $\mathrm{V}\Sigma^i\mathrm{P}$ et $\mathrm{V}\Pi^i\mathrm{P}$, grâce à des grosses sommes et des gros produits. Nous nous intéressons alors à des problèmes d'effondrement et de séparation de la hiérarchie, puis, grâce à un circuit de séparation d'hyperplans, obtenons un résultat de transfert vers la complexité par machines de Turing algébriques.

Nous espérons que cette hiérarchie puisse créer un cadre propice à l'étude de questions fondamentales faisant intervenir les gros produits, par exemple de savoir si la factorielle est facilement calculable (*i.e.* s'il existe une constante c telle que pour tout k, il existe un circuit calculant $k!$ de taille $\leq(\log k)^c$). Ou par exemple la τ-conjecture de [3], qui stipule qu'il existe une constante c telle que le nombre de racines entières d'un polynôme $f\in\mathbf{Z}[t]$ est majoré par $(1+\tau(f))^c$, où $\tau(f)$ représente la taille du plus petit circuit calculant f (cette conjecture implique P $\neq$ NP sur $\mathbf{C}$).

Dans le second chapitre, l'approche est différente. Nous nous penchons cette fois sur la complexité par machine de Turing booléenne de la manipulation de polynômes donnés par des circuits algébriques. En effet, il peut être pratique de coder un polynôme par un circuit algébrique : en quelque sorte, on donne une méthode de calcul du polynôme plutôt que d'expliciter chaque monôme un à un. Si l'on compte utiliser ce codage, il faut être capable de travailler efficacement sur les circuits algébriques. *Via* deux problèmes, nous montrons que la manipulation n'est pas toujours aisée ; elle dépend notamment du «degré formel complet» du circuit (le degré du polynôme obtenu lorsque l'on remplace les constantes par de nouvelles variables). Par exemple, le calcul du degré d'un polynôme se fait efficacement lorsque le degré formel complet du circuit est polynomial. En revanche, nous ne connaissons pas d'algorithme efficace dans le cas où le degré formel complet est quelconque. Pour le second problème, où il s'agit de calculer le coefficient d'un monôme dans le circuit, on montre une borne inférieure qui élimine l'espoir de résoudre ce problème efficacement.

Avant d'aborder le cœur de notre étude, nous rappelons quelques notions classiques de complexité dont nous nous servons dans ce rapport (voir par exemple [16] ou [8] pour plus de détails).

P (resp. NP) est la classe des problèmes décidables en temps polynomial sur une machine de Turing déterministe (resp. non déterministe). Si L est un langage, on note P^{L} (resp. NP^{L}) la classe des langages reconnus en temps polynomial déterministe (resp. non déterministe) par une machine avec oracle L. De même, si $\mathcal{C}$ est une classe de complexité, on note $\mathrm{P}^{\mathcal{C}}$ (resp. $\mathrm{NP}^{\mathcal{C}}$) l'union pour tout $L \in \mathcal{C}$ des P^{L} (resp. NP^{L}).

On définit par récurrence sur $i \geq 0$ les classes Σ_i^p comme suit : $\Sigma_0^p = \mathrm{P}$ et $\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p}$. La *hiérarchie polynomiale* est définie par $\mathrm{PH} = \cup_{i\geq 0}\Sigma_i^p$. La classe PSPACE est l'ensemble des langages décidables en espace polynomial.

La classe UP est l'ensemble des langages reconnus en temps polynomial par machine de Turing ayant au plus un chemin acceptant : $L \in \mathrm{UP}$ s'il existe $A \in \mathrm{P}$ et un polynôme $p(n)$ tels que

$$x \in L \Rightarrow \mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} = 1, \text{ et } x \notin L \Rightarrow \mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} = 0.$$

La classe $\oplus\mathrm{P}$ compte la parité du nombre de chemins acceptants d'une machine non déterministe fonctionnant en temps polynomial : $L \in \oplus\mathrm{P}$ s'il existe $A \in \mathrm{P}$ et un polynôme $p(n)$ tels que

$$x \in L \Leftrightarrow \mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} \text{ impair.}$$

Plus généralement, pour $k \geq 2$ on définit la classe $\mathrm{Mod}_k\mathrm{P}$ par l'ensemble des langages L vérifiant

$$x \in L \Leftrightarrow \mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} \not\equiv 0 \bmod k.$$

Pour k premier, $\mathrm{Mod}_k\mathrm{P}$ est close par complément ([1]), donc on peut prendre $\equiv 0$ dans la définition.

Enfin, les classes Co-RP et BPP sont probabilistes. Co-RP est l'ensemble des langages L tels qu'il existe un machine probabiliste qui ne se trompe pas sur les entrées qui sont dans L, et qui a une faible probabilité d'erreur pour celles qui ne sont pas dans L. Formellement, un langage L est dans Co-RP s'il existe $A \in \mathrm{P}$ et un polynôme $p(n)$ tels que pour tout $x \in \{0,1\}^*$,
- si $x \in L$, alors $\mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} = 0$;
- si $x \notin L$, alors $\mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} \geq 2^{p(|x|)-1}$.

BPP est défini de la même façon mais on autorise les erreurs des deux côtés :
- si $x \in L$, alors $\mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} \geq 3/4.2^{p(|x|)}$;
- si $x \notin L$, alors $\mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\} \leq 1/4.2^{p(|x|)}$.

Nous aurons aussi besoin de classes de fonctions. FP (resp. FPSPACE) est l'ensemble des fonctions calculables en temps (resp. espace) polynomial. #P est la classe des fonctions qui comptent le nombre de chemins acceptants d'une machine de Turing non déterministe fonctionnant en temps polynomial. Plus formellement, $f : \{0,1\}^* \to \mathbf{N}$ est dans #P s'il existe un langage $A \in \mathrm{P}$ et un polynôme $p(n)$ tel que

$$f(x) = \mathrm{card}\{y \in \{0,1\}^{p(|x|)} : (x,y) \in A\}.$$

GapP est la classe des fonctions de $\{0,1\}^*$ dans $\mathbf{Z}$ qui sont la différence de deux fonctions #P.

On rappelle quelques inclusions ($\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$ constitue le théorème de Toda [19]) :

$$\mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{PSPACE}, \text{ et } \mathrm{P} \subseteq \text{Co-RP} \subseteq \mathrm{BPP} \subseteq \Sigma_2^p.$$

Enfin, lorsque l'on a une classe de complexité $\mathcal{C}$, on définit sa version non uniforme $\mathcal{C}/\mathrm{poly}$ comme étant l'ensemble des langages L tels qu'il existe un langage $L' \in \mathcal{C}$ et une fonction de conseil $u : \mathbf{N} \to \{0,1\}^*$ (*i.e.* $|u(n)|$ est polynomialement borné en n) vérifiant : $x \in L$ si et seulement si $(x, u(|x|)) \in L'$. On a la même notion pour une classe de fonctions $\mathcal{F}$: une fonction f est dans $\mathcal{F}/\mathrm{poly}$ s'il existe $g \in \mathcal{F}$ et une fonction de conseil u telles que $f(x) = g(x, u(|x|))$.

Rappelons enfin quelques résultats sur le permanent. Le permanent d'une matrice est défini similairement au déterminant, à ceci près qu'on ne prend pas en compte la signature de la permutation. Plus précisément, si $M = (m_{i,j})$ est une matrice carrée de taille n, on définit

$$\mathrm{per}(M) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} m_{i,\sigma(i)},$$

où $\mathcal{S}_n$ est l'ensemble des permutations de $\{1, \ldots, n\}$. Malgré cette ressemblance avec le déterminant, nous ne savons pas calculer le permanent efficacement (alors que le déterminant se calcule en temps polynomial, par exemple par réduction de Gauss). Cette difficulté a été capturée dans le résultat de [22] : le calcul du permanent est #P-complet (même pour les matrices à coefficients dans $\{0,1\}$). De même, pour la notion de complétude que nous verrons dans le premier chapitre, la suite (P_n) des permanents de matrices carrées de taille n est VNP-complète.

Chapitre 1

Une hiérarchie généralisant le modèle de Valiant

1.1 Introduction

Dans le cadre de la théorie de Valiant, on s'intéresse aux suites de polynômes «facilement calculables» (classe VP), et «facilement définissables» (classe VNP). La première classe correspond aux suites de polynômes de degré polynomialement borné, calculables par des circuits algébriques de taille polynomiale. Dans la seconde, on s'autorise de sommer un nombre simplement exponentiel de tels polynômes. Dans sa thèse [15], Guillaume Malod étudie les classes analogues sans restriction sur le degré des polynômes. Il étudie également les classes dans lesquelles on ne s'autorise pas de constante arbitraire. Dans ce qui suit, nous nous plaçons dans ce cadre : nous n'autorisons pas de constante arbitraire, et ne faisons aucune restriction sur le degré des polynômes calculés.

La puissance obtenue dans VNP en effectuant des grosses sommes ne permet toutefois pas clairement de calculer la factorielle, par exemple. Cependant, ce calcul apparaît fréquemment dans ce domaine : par exemple, dans [15, section 5.5], ainsi que dans [3] (en lien avec la «τ-conjecture»). Ce problème est étudié dans [13]. On ne sait pas si l'on peut calculer efficacement la factorielle. Il semble alors naturel d'envisager des classes de polynômes définis par de gros produits (plutôt que par de grosses sommes), dans lesquelles $(2^n!)$ se calcule facilement. C'est ce qui nous amène à étudier la classe $\mathrm{V\Pi P}$. De même, on pourrait faire des sommes de produits, des produits de sommes de produits, etc. Ainsi, on définit une hiérarchie. C'est l'objet du début de ce chapitre.

Après avoir vu quelques propriétés élémentaires de clôture de ces classes, nous exhibons des suites de polynômes complètes pour chacune d'elles. Puis nous étudions sous quelles hypothèses ces classes coïncident, ou sont distinctes. Enfin, nous exhibons un circuit de séparation d'hyperplans, avec des portes de tests $\mathrm{V\Pi P}$. Ces gros produits nous permettent d'effectuer des recherches dichotomiques. Nous en déduisons un résutat de transfert portant sur les classes algébriques P et NP non uniformes (dans le cadre du modèle BSS [5]), sous l'hypothèse que les gros produits sont calculables facilement.

1.2 Définition de la hiérarchie

Nous rappelons la définition ([15]) de la classe $\mathrm{VP}^0_{\mathrm{nb}}$ des suites de polynômes calculables par des circuits de taille polynomiale, sans constante, et de degré quelconque («nb» = non borné) :

1.2.1 Définition La classe $\mathrm{VP}^0_{\mathrm{nb}}$ est l'ensemble des suites de polynômes calculables par des circuits algébriques de taille polynomiale. Les seules constantes autorisées sont -1, 0 et 1.

La classe $\mathrm{VP}^0_{\mathrm{nb}}$ constitue le premier niveau de notre hiérarchie. Nous le désignerons indifféremment par $\mathrm{V\Sigma^0P}$ ou $\mathrm{V\Pi^0P}$. Notons que l'exposant 0 signifie qu'il s'agit du niveau zéro de la hiérarchie, alors que dans le cas de $\mathrm{VP}^0_{\mathrm{nb}}$, il signale l'absence de constante arbitraire. Pour éviter toute confusion, nous utiliserons de préférence $\mathrm{V\Sigma^0P}$, puis $\mathrm{V\Pi^0P}$, mais plus $\mathrm{VP}^0_{\mathrm{nb}}$.

Nous définissons maintenant par récurrence les autres niveaux de la hiérarchie. Pour $k > 0$, la classe $\mathrm{V}\Sigma^k\mathrm{P}$ est l'ensemble des suites de polynômes $(f_n(x_1, \ldots, x_{u(n)}))$ telles qu'il existe une suite de polynômes $(g_n(x_1, \ldots, x_{u(n)}, y_1, \ldots, y_{v(n)})) \in \mathrm{V}\Pi^{k-1}\mathrm{P}$, vérifiant pour tout n :

$$f_n(x_1, \ldots, x_{u(n)}) = \sum_{\bar{\epsilon} \in \{0,1\}^{v(n)}} g_n(x_1, \ldots, x_{u(n)}, \bar{\epsilon}).$$

Les fonctions $u(n)$ et $v(n)$ sont polynomialement bornées.

De manière analogue, la classe $\mathrm{V}\Pi^k\mathrm{P}$ est définie par des produits. Plus précisément, pour $k > 0$ la classe $\mathrm{V}\Pi^k\mathrm{P}$ est l'ensemble des suites de polynômes $(f_n(x_1, \ldots, x_{u(n)}))$ telles qu'il existe une suite de polynômes $(g_n(x_1, \ldots, x_{u(n)}, y_1, \ldots, y_{v(n)})) \in \mathrm{V}\Sigma^{k-1}\mathrm{P}$, vérifiant pour tout n :

$$f_n(x_1, \ldots, x_{u(n)}) = \prod_{\bar{\epsilon} \in \{0,1\}^{v(n)}} g_n(x_1, \ldots, x_{u(n)}, \bar{\epsilon}).$$

La classe $\mathrm{V}\Sigma^k\mathrm{P}$ représente donc les polynômes qui sont une somme, de produit, de somme, etc. (k fois), de polynômes dans $\mathrm{V}\Sigma^0\mathrm{P}$. De même, la classe $\mathrm{V}\Pi^k\mathrm{P}$ représente les polynômes qui sont un produit, de somme, de produit, etc. (k fois), de polynômes dans $\mathrm{V}\Sigma^0\mathrm{P}$. On retrouve l'esprit de la hiérarchie polynomiale en complexité booléenne, où le quantificateur $\exists$ est remplacé par la somme $\sum$, et le quantificateur $\forall$ est remplacé par le produit $\prod$. Bien entendu, on ne reconnaît pas des langages avec nos circuits algébriques, mais on calcule des polynômes. De manière évidente, on a les inclusions suivantes pour tout $i \geq 0$:

$$(\mathrm{V}\Sigma^i\mathrm{P} \cup \mathrm{V}\Pi^i\mathrm{P}) \subseteq \mathrm{V}\Pi^{i+1}\mathrm{P} \text{ et } (\mathrm{V}\Sigma^i\mathrm{P} \cup \mathrm{V}\Pi^i\mathrm{P}) \subseteq \mathrm{V}\Sigma^{i+1}\mathrm{P}.$$

Pour plus de lisibilité, on adoptera la notation $\mathrm{V}\Sigma\mathrm{P}$ et $\mathrm{V}\Pi\mathrm{P}$ respectivement pour $\mathrm{V}\Sigma^1\mathrm{P}$ et $\mathrm{V}\Pi^1\mathrm{P}$. De même, s'il n'y a pas d'ambiguïté, on remplacera souvent $\bar{\epsilon}$ par un entier i (sous-entendu, écrit en binaire), et $\bar{x}$ par x tout simplement. Les chiffres binaires de i seront notés i_j ($j \geq 0$). Ainsi, on écrira par exemple

$$f_n(x) = \sum_{i=0}^{2^{p(n)}-1} g_n(i, x), \quad \text{à la place de} \quad f_n(\bar{x}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(n)}} g_n(\bar{\epsilon}, \bar{x}).$$

1.3 Premières propriétés

Nous étudions dans cette partie les propriétés de base des classes $\mathrm{V}\Sigma^k\mathrm{P}$ et $\mathrm{V}\Pi^k\mathrm{P}$: clôture par produit, somme, etc. Nous commençons par une série de lemmes qui préparent la démonstration de la proposition 1.3.2.1.

1.3.1 Lemmes techniques

Les deux remarques suivantes sont immédiates par définition des classes $\mathrm{V}\Sigma^k\mathrm{P}$ et $\mathrm{V}\Pi^k\mathrm{P}$.

1.3.1.1 Remarque Soit $(f_n(i, x))$ dans $\mathrm{V}\Sigma^k\mathrm{P}$ pour un certain $k \geq 1$. Soit $p(n)$ un polynôme. Alors la suite constituée des polynômes suivants est dans $\mathrm{V}\Sigma^k\mathrm{P}$:

$$g_n(x) = \sum_{i=0}^{2^{p(n)}-1} f_n(i, x).$$

1.3.1.2 Remarque Soit $(f_n(i, x))$ dans $\mathrm{V}\Pi^k\mathrm{P}$ pour un certain $k \geq 1$. Soit $p(n)$ un polynôme. Alors la suite constituée des polynômes suivants est dans $\mathrm{V}\Pi^k\mathrm{P}$:

$$g_n(x) = \prod_{i=0}^{2^{p(n)}-1} f_n(i, x).$$

Les deux prochains lemmes étudient les propriétés de clôture des classes $V\Sigma^k P$ et $V\Pi^k P$ par multiplication ou addition avec une fonction $V\Sigma^0 P$.

1.3.1.3 Lemme Pour tout $k \geq 0$, et tout (f_n) dans $V\Sigma^0 P$, on a :

(i) pour tout (g_n) dans $V\Sigma^k P$, $(f_n g_n)$ est dans $V\Sigma^k P$;

(ii) pour tout (g_n) dans $V\Pi^k P$, $(f_n g_n)$ est dans $V\Pi^k P$.

Preuve Par récurrence. Le résultat est clair pour $k = 0$. Soit $k > 0$.

(i) (g_n) est dans $V\Sigma^k P$, donc s'écrit sous la forme $g_n(x) = \sum_{i=0}^{2^{p(n)}-1} h_n(i, x)$, où (h_n) est dans $V\Pi^{k-1} P$. Par induction, $((i, x) \mapsto f_n(x)h_n(i, x))$ est dans $V\Pi^{k-1} P$, et

$$f_n g_n(x) = \sum_{i=0}^{2^{p(n)}-1} f_n(x)h_n(i, x),$$

ce qui conclut.

(ii) (g_n) est dans $V\Pi^k P$, donc s'écrit sous la forme $g_n(x) = \prod_{i=0}^{2^{p(n)}-1} h_n(i, x)$, où (h_n) est dans $V\Sigma^{k-1} P$. On définit

$$h'_n(i, x) = ((\prod_{j=0}^{p(n)-1} i_j) f_n(x) + 1 - \prod_{j=0}^{p(n)-1} i_j) h_n(i, x),$$

où i_j est le j-ème chiffre binaire de i. On a : $h'_n(i, x) = h_n(i, x)$ lorsque $i < 2^{p(n)} - 1$, et $h'_n(i, x) = f_n(x)h_n(i, x)$ lorsque $i = 2^{p(n)} - 1$. Ainsi,

$$\prod_{i=0}^{2^{p(n)}-1} h'_n(i, x) = f_n(x)g_n(x).$$

De plus, (h'_n) est dans $V\Sigma^{k-1} P$: en effet, $((\prod_{j=0}^{p(n)-1} i_j) f_n(x) + 1 - \prod_{j=0}^{p(n)-1} i_j)$ est dans $V\Sigma^0 P$, et (h_n) dans $V\Sigma^{k-1} P$, donc par hypothèse d'induction le produit est dans $V\Sigma^{k-1} P$.

□

1.3.1.4 Lemme Pour tout $k \geq 0$, et tout (f_n) dans $V\Sigma^0 P$, on a :

(i) pour tout (g_n) dans $V\Sigma^k P$, $(f_n + g_n)$ est dans $V\Sigma^k P$;

(ii) pour tout (g_n) dans $V\Pi^k P$, $(f_n + g_n)$ est dans $V\Sigma^{k+1} P$.

Preuve Par récurrence sur k. Pour $k = 0$, le résultat est clair. Soit $k > 0$.

(i) g_n s'écrit sous la forme $g_n(x) = \sum_{i=0}^{2^{p(n)}-1} h_n(i, x)$, avec (h_n) dans $V\Pi^{k-1} P$. On pose (pour $j \in \{0, 1\}$) :

$$h'_n(j, i, x) = j h_n(i, x) + (1 - j)(\prod_{k=0}^{p(n)-1} i_k) f_n(x).$$

Alors h'_n vérifie $h'_n(1, i, x) = h_n(i, x)$ (pour tout i), $h'_n(0, i, x) = 0$ pour $i < 2^{p(n)} - 1$, et $h'_n(0, 2^{p(n)} - 1, x) = f_n(x)$. Ainsi,

$$\sum_{j=0}^{1} \sum_{i=0}^{2^{p(n)}-1} h'_n(j, i, x) = (f_n + g_n)(x).$$

De plus, (h'_n) est dans $V\Sigma^k P$: par le lemme 1.3.1.3, $(j h_n(i, x))$ est dans $V\Pi^{k-1} P$, et comme $((1 - j)(\prod_{k=0}^{p(n)-1} i_k) f_n(x))$ est dans $V\Sigma^0 P$, par induction on conclut que la somme de ces deux termes est dans $V\Sigma^k P$. Ainsi, $(f_n + g_n)$ est dans $V\Sigma^k P$.

(ii) g_n s'écrit sous la forme $g_n(x) = \prod_{i=0}^{2^{p(n)}-1} h_n(i,x)$, avec (h_n) dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$. On pose (pour $j \in \{0,1\}$) :

$$h'_n(j,i,x) = jh_n(i,x) + (1-j)[(\prod_{k=0}^{p(n)-1} i_k)f_n(x) + 1 - (\prod_{k=0}^{p(n)-1} i_k)].$$

Alors h'_n vérifie $h'_n(1,i,x) = h_n(i,x)$ (pour tout i), $h'_n(0,i,x) = 1$ pour $i < 2^{p(n)} - 1$, et $h'_n(0, 2^{p(n)} - 1, x) = f_n(x)$. Ainsi,

$$\sum_{j=0}^{1} \prod_{i=0}^{2^{p(n)}-1} h'_n(j,i,x) = (f_n + g_n)(x).$$

De plus, (h'_n) est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, puisque par le lemme 1.3.1.3, $(jh_n(i,x))$ est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, et comme $((1-j)[(\prod_{k=0}^{p(n)-1} i_k)f_n(x) + 1 - (\prod_{k=0}^{p(n)-1} i_k)])$ est dans $\mathrm{V}\Sigma^{0}\mathrm{P}$, par induction on conclut que la somme de ces deux termes est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$. Ainsi, $(f_n + g_n)$ est dans $\mathrm{V}\Sigma^{k+1}\mathrm{P}$. □

Les deux lemmes qui suivent permettent de supposer que pour deux fonctions quelconques de $\mathrm{V}\Sigma^{k}\mathrm{P}$ (resp. $\mathrm{V}\Pi^{k}\mathrm{P}$), les sommations (resp. produits) varient sur le même intervalle. En effet, ils affirment que l'on peut augmenter la borne supérieure des sommations (resp. produits), tout en restant dans $\mathrm{V}\Sigma^{k}\mathrm{P}$ (resp. $\mathrm{V}\Pi^{k}\mathrm{P}$). Cela nous servira dans la preuve de la proposition 1.3.2.1.

1.3.1.5 Lemme Soient p et q des polynômes vérifiant $p(n) \leq q(n)$ pour tout n. Soient $k \geq 1$ un entier et (f_n) une suite de $\mathrm{V}\Sigma^{k}\mathrm{P}$, qui s'exprime par

$$f_n(x) = \sum_{i=0}^{2^{p(n)}-1} g_n(i,x),$$

où (g_n) est dans $\mathrm{V}\Pi^{k-1}\mathrm{P}$. Alors il existe une suite (h_n) de $\mathrm{V}\Pi^{k-1}\mathrm{P}$ telle que

$$f_n(x) = \sum_{i=0}^{2^{q(n)}-1} h_n(i,x).$$

(L'indice i varie maintenant entre 0 et $2^{q(n)} - 1$, et plus entre 0 et $2^{p(n)} - 1$)

Preuve On définit h_n comme suit :

$$h_n(i,x) = (\prod_{j=p(n)}^{q(n)-1} (1 - i_j))g_n(i,x),$$

où i_j désigne le j-ème chiffre binaire de i. La fonction h_n vérifie alors $h_n(i,x) = 0$ si $i \geq 2^{p(n)}$, et $h_n(i,x) = g_n(i,x)$ sinon, ce qui implique que $f_n(x) = \sum_{i=0}^{2^{q(n)}-1} h_n(i,x)$. Bien entendu, (h_n) est dans $\mathrm{V}\Pi^{k-1}\mathrm{P}$: on multiplie en effet $(\prod_{j=p(n)}^{q(n)-1}(1-i_j))$, de $\mathrm{V}\Sigma^{0}\mathrm{P}$, par (g_n), de $\mathrm{V}\Pi^{k-1}\mathrm{P}$, donc (h_n) est dans $\mathrm{V}\Pi^{k-1}\mathrm{P}$ par le lemme 1.3.1.3. □

1.3.1.6 Lemme Soient p et q des polynômes vérifiant $p(n) \leq q(n)$ pour tout n. Soient $k \geq 1$ un entier et (f_n) une suite de $\mathrm{V}\Pi^{k}\mathrm{P}$, qui s'exprime par

$$f_n(x) = \prod_{i=0}^{2^{p(n)}-1} g_n(i,x),$$

où (g_n) est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$. Alors il existe une suite (h_n) de $\mathrm{V}\Sigma^{k-1}\mathrm{P}$ telle que

$$f_n(x) = \prod_{i=0}^{2^{q(n)}-1} h_n(i,x).$$

(L'indice i varie maintenant entre 0 et $2^{q(n)}-1$, et plus entre 0 et $2^{p(n)}-1$)

Preuve On définit h_n comme suit :

$$h_n(i,x) = (\prod_{j=p(n)}^{q(n)-1}(1-i_j))g_n(i,x) + 1 - \prod_{j=p(n)}^{q(n)-1}(1-i_j),$$

où i_j désigne le j-ème chiffre binaire de i. La fonction h_n vérifie alors $h_n(i,x)=1$ si $i \geq 2^{p(n)}$, et $h_n(i,x)=g_n(i,x)$ sinon, ce qui implique que $f_n(x)=\prod_{i=0}^{2^{q(n)}-1} h_n(i,x)$. Bien entendu, (h_n) est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$: en effet, on multiplie $(\prod_{j=p(n)}^{q(n)-1}(1-i_j))$, de $\mathrm{V}\Sigma^0\mathrm{P}$, par (g_n), de $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, ce qui reste $\mathrm{V}\Sigma^{k-1}\mathrm{P}$ par le lemme 1.3.1.3 ; puis on ajoute une fonction $\mathrm{V}\Sigma^0\mathrm{P}$: on reste dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$ par le lemme 1.3.1.4. D'où le résultat. □

1.3.2 Résultats principaux de stabilité

La proposition suivante résume différentes propriétés de clôture des classes $\mathrm{V}\Sigma^k\mathrm{P}$ et $\mathrm{V}\Pi^k\mathrm{P}$.

1.3.2.1 Proposition

(i) Pour tout k, la classe $\mathrm{V}\Sigma^k\mathrm{P}$ est close par :
  - addition ;
  - multiplication ;
  - opposé ($x \mapsto -x$).

(ii) En ce qui concerne la classe $\mathrm{V}\Pi^k\mathrm{P}$, pour tout k on a :
  - la somme de deux suites $\mathrm{V}\Pi^k\mathrm{P}$ est $\mathrm{V}\Sigma^{k+1}\mathrm{P}$;
  - le produit de deux suites $\mathrm{V}\Pi^k\mathrm{P}$ est $\mathrm{V}\Pi^k\mathrm{P}$;
  - l'opposé d'une suite $\mathrm{V}\Pi^k\mathrm{P}$ est $\mathrm{V}\Pi^k\mathrm{P}$.

Preuve On procède par récurrence sur k. Les résultats sont clairs pour $k=0$. Soit $k>0$.

(i) Soient (f_n) et (g_n) des suites de fonctions de $\mathrm{V}\Sigma^k\mathrm{P}$. Par le lemme 1.3.1.5, on peut supposer qu'elles s'expriment par

$$f_n(x) = \sum_{i=0}^{2^{p(n)}-1} f'_n(i,x) \text{ et } g_n(x) = \sum_{i=0}^{2^{p(n)}-1} g'_n(i,x),$$

pour un certain polynôme p, et (f'_n) et (g'_n) dans $\mathrm{V}\Pi^{k-1}\mathrm{P}$.

  - Pour l'addition : par récurrence, on sait que $(f'_n+g'_n)$ est dans $\mathrm{V}\Sigma^k\mathrm{P}$ (somme de deux suites $\mathrm{V}\Pi^{k-1}\mathrm{P}$). Donc la suite composée des polynômes suivants est dans $\mathrm{V}\Sigma^k\mathrm{P}$:

$$(f_n+g_n)(x) = \sum_{i=0}^{2^{p(n)}-1}(f'_n+g'_n)(i,x)$$

  - Pour la multiplication : par récurrence, on sait que $((i,j,x)\mapsto f'_n(i,x)g'_n(j,x))$ est dans $\mathrm{V}\Pi^{k-1}\mathrm{P}$ (produit de deux suites $\mathrm{V}\Pi^{k-1}\mathrm{P}$). Or

$$f_ng_n(x) = \sum_{i=0}^{2^{p(n)}-1}\sum_{j=0}^{2^{p(n)}-1} f'_n(i,x)g'_n(j,x),$$

    donc (f_ng_n) est dans $\mathrm{V}\Sigma^k\mathrm{P}$.
  - Pour l'opposé : la suite $(x \mapsto -1)$ est dans $\mathrm{V}\Sigma^0\mathrm{P}$, donc par le lemme 1.3.1.3, $(-f_n)$ est dans $\mathrm{V}\Sigma^k\mathrm{P}$.

(ii) Soient (f_n) et (g_n) des suites de fonctions de $\mathrm{V}\Pi^k\mathrm{P}$. Par le lemme 1.3.1.6, on peut supposer qu'elles s'expriment par

$$f_n(x) = \prod_{i=0}^{2^{p(n)}-1} f'_n(i,x) \text{ et } g_n(x) = \prod_{i=0}^{2^{p(n)}-1} g'_n(i,x),$$

pour un certain polynôme p, et (f'_n) et (g'_n) dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$.

– Pour l'addition : soit
$$h'_n(j,i,x) = jf'_n(i,x) + (1-j)g'_n(i,x),$$
pour $j \in \{0,1\}$. Par le lemme 1.3.1.3, il s'agit d'une somme de deux suites $\mathrm{V}\Sigma^{k-1}\mathrm{P}$. Par induction, (h'_n) est donc dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, et on a :
$$(f_n + g_n)(x) = \sum_{j=0}^{1} \prod_{i=0}^{2^{p(n)}-1} h'_n(j,i,x),$$
donc $(f_n + g_n)$ est dans $\mathrm{V}\Sigma^{k+1}\mathrm{P}$.

– Pour la multiplication : on pose comme précédemment
$$h'_n(j,i,x) = jf'_n(i,x) + (1-j)g'_n(i,x).$$
Alors (h'_n) est dans $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, donc la suite des polynômes suivants est dans $\mathrm{V}\Pi^k\mathrm{P}$:
$$f_n g_n(x) = \prod_{j=0}^{1} \prod_{i=0}^{2^{p(n)}-1} h'_n(j,i,x).$$

– Pour l'opposé : on applique le lemme 1.3.1.3.

□

Nous voyons donc que les classes ainsi définies vérifient des propriétés naturelles de clôture, mais il faut toutefois être prudent : la somme de deux suites $\mathrm{V}\Pi^k\mathrm{P}$ n'est pas clairement dans $\mathrm{V}\Pi^k\mathrm{P}$, mais dans $\mathrm{V}\Sigma^{k+1}\mathrm{P}$.

Nous étudions maintenant l'ensemble sur lequel nous pouvons faire les sommations ou les produits. D'après la définition, il s'agit d'un intervalle de la forme $0 \le i < 2^{p(n)}$, où $p(n)$ est un polynôme. La proposition suivante permet de considérer des ensembles plus compliqués, c'est-à-dire d'effectuer des sommes (ou des produits) de la forme
$$\sum_{i \in I} f_n(i,x)$$
pour certains ensembles I. Lorsque A est un langage sur $\Sigma = \{0,1\}$, la notation $A^{=m}$ désigne l'ensemble des mots de A de taille m, *i.e.* $A \cap \Sigma^m$. La proposition suivante est énoncée pour les produits, mais bien sûr elle est aussi vraie pour les sommes.

1.3.2.2 Proposition Soient $k > 0$, et $p(n)$ un polynôme. Soit $(f_n(i,x))$ une suite de $\mathrm{V}\Sigma^{k-1}\mathrm{P}$, où i, uple de variables, est de taille $t_i(n) \le p(n)$ (intuitivement, i va correspondre à un mot sur $\{0,1\}$), et x un uple de taille $|x| \le p(n)$. Soit $A \in \mathrm{P/poly}$ un langage. Alors la suite $(g_n(x))$ constituée des polynômes suivants est dans $\mathrm{V}\Pi^k\mathrm{P}$:
$$g_n(x) = \prod_{i \in A^{=t_i(n)}} f_n(i,x).$$

Preuve Puisque $A \in \mathrm{P/poly}$, il existe une suite (C_n) de circuits booléens décidant A, de taille polynomiale. En simulant les opérations booléennes par des polynômes, de manière classique, on obtient une suite de fonctions $(c_n(i))$ dans $\mathrm{V}\Sigma^0\mathrm{P}$, où i est de taille n, telle que pour toute entrée booléenne i,
$$c_n(i) = \begin{cases} 0 & \text{si } i \notin A \\ 1 & \text{sinon.} \end{cases}$$
La suite des fonctions (prenant les mêmes arguments que (f_n))
$$h_n(i,x) = c_{t_i(n)}(i) f_n(i,x) + 1 - c_{t_i(n)}(i)$$

est dans $V\Sigma^{k-1}P$ par les lemmes 1.3.1.3 et 1.3.1.4 ($(c_{t_i(n)})$ est $V\Sigma^0P$ puisque $t_i(n)$ est polynomialement bornée). Donc on a

$$g_n(x) = \prod_{i \in A^{=t_i(n)}} f_n(i,x) = \prod_{i=0}^{2^{t_i(n)}-1} h_n(i,x)$$

et (g_n) est $V\Pi^kP$. □

1.4 Polynômes complets

Dans cette section, on s'intéresse à des polynômes complets pour les classes que l'on vient de définir. Dans un premier temps, nous montrons qu'elles sont stables par projection, puis nous exhibons un polynôme complet pour chacune d'elles.

1.4.1 Stabilité par projection

On reprend la notion de projection (non bornée) de la thèse de Guillaume Malod :

1.4.1.1 Définition Soient deux suites de polynômes $(f_n(x_1,\ldots,x_{u(n)}))$ et $(g_n(y_1,\ldots,y_{v(n)}))$. On dit que (g_n) est une *projection de* (f_n) *via les fonctions* $s(n)$ *et* $t(n)$ lorsque :
- $s(n)$ et $t(n)$ sont polynomialement bornées ;
- $g_n(y_1,\ldots,y_{v(n)}) = f_{t(n)}(a_1,\ldots,a_{u(t(n))})$, où a_i est soit l'une des variables y_j, soit un entier calculable par un circuit algébrique de taille $\leq s(n)$ à partir des constantes -1, 0 et 1.

On dit que (g_n) est une *projection* de (f_n) lorsqu'il existe $s(n)$ et $t(n)$ tels que (g_n) est une projection de (f_n) *via* $s(n)$ et $t(n)$.

Montrons que pour tout i, les classes $V\Sigma^iP$ et $V\Pi^iP$ sont stables par projection. Raisonnons pour cela par récurrence sur i.

Pour $i = 0$: soit (g_n) une projection de $(f_n) \in V\Sigma^0P$, *via* les fonctions $s(n)$ et $t(n)$. Alors pour tout n, le polynôme g_n est obtenu à partir de $f_{t(n)}$ en remplaçant certaines variables par d'autres ou par des entiers relatifs calculables par un circuit de taille inférieure à $s(n)$. Si l'on effectue ces remplacements dans le circuit calculant $f_{t(n)}$, on obtient un circuit $V\Sigma^0P$ calculant g_n.

Pour $i > 0$, soit (g_n) une projection de $(f_n) \in V\Sigma^iP$, *via* les fonctions $s(n)$ et $t(n)$. Puisque $(f_n) \in V\Sigma^iP$, le polynôme $f_{t(n)}$ est de la forme

$$f_{t(n)}(X_1,\ldots,X_{u(n)}) = \sum_{\bar{\epsilon}\in\{0,1\}^{v(n)}} f'_n(X_1,\ldots,X_{u(n)},\epsilon_1,\ldots,\epsilon_{v(n)})$$

pour une certaine suite (f'_n) de $V\Pi^{i-1}P$. Supposons que g_n soit obtenu à partir de $f_{t(n)}$ en remplaçant X_j par a_j, où a_j est soit une variable, soit un entier relatif calculable par un circuit de taille inférieure à $s(n)$. En effectuant les mêmes changements dans les variables X_j de f'_n, on obtient un polynôme g'_n de $V\Pi^{i-1}P$, puisque par récurrence $V\Pi^{i-1}P$ est stable par projection. Ainsi, (g_n) est dans $V\Sigma^iP$, ce qui conclut (le cas de $V\Pi^iP$ est identique).

1.4.2 Des polynômes $V\Sigma^iP$- et $V\Pi^iP$-complets

La notion de projection nous permet de définir une notion de complétude.

1.4.2.1 Définition Soit $\mathcal{C}$ une classe de suites de polynômes. On dit qu'une suite (f_n) est *complète* pour $\mathcal{C}$ (ou encore $\mathcal{C}$-complète) lorsque les deux conditions suivantes sont vérifiées :
- $(f_n) \in \mathcal{C}$;
- toute suite $(g_n) \in \mathcal{C}$ est une projection de (f_n).

En suivant la technique de la thèse de Guillaume Malod pour $V\Sigma P$, on va construire des suites de polynômes complètes pour chaque classe $V\Sigma^iP$ et $V\Pi^iP$. Rappelons l'expression de $(G_n(\bar{a},\bar{b},\bar{y}))$, la suite de polynômes $V\Sigma^0P$-complète exhibée dans la thèse (notée (G_n^n) dans celle-ci) : il s'agit de simuler par niveaux un circuit générique. Le premier niveau consiste en les variables et la constante 1 : on définit

$$G^n_{-n} = 1, G^n_{-n-1} = x_1, \dots, G^n_0 = x_n.$$

Puis on calcule tous les produits et sommes possibles du niveau $l+1$, en ajoutant de nouvelles variables $a_{l+1,i}$ et $b_{l+1,i}$, par :

$$G^n_{l+1} = \left(\sum_{i=-n}^{l} a_{l+1,i} G^n_i\right)\left(\sum_{i=-n}^{l} b_{l+1,i} G^n_i\right).$$

On note alors $G_n = G^n_n$. La suite (G_n) est $\mathrm{V\Sigma^0P}$-complète (puisqu'on a simulé un circuit générique) : on pourra en trouver une preuve dans [15].

Dans la notation $(G_n(\bar{a}, \bar{b}, \bar{y}))$, la taille de $\bar{y}$ est n. Dans la thèse, il est montré que toute suite (f_n) de $\mathrm{V\Sigma^0P}$ s'écrit sous la forme

$$f_n(x_1, \dots, x_{p(n)}) = G_{q(n)}(\bar{u}, \bar{v}, w_1, \dots, w_{q(n)-p(n)}, x_1, \dots, x_{p(n)}),$$

où les u_j, v_j et w_j ont pour valeur -1, 0, 1 ou 2.

Nous avons ainsi notre suite $\mathrm{V\Sigma^0P}$-complète (et aussi $\mathrm{V\Pi^0P}$-complète puisque ces deux classes sont les mêmes). Construisons par récurrence une suite de polynômes $(C\Sigma^{(i)}_n)$ qui soit $\mathrm{V\Sigma^iP}$-complète, et $(C\Pi^{(i)}_n)$ qui soit $\mathrm{V\Pi^iP}$-complète, telle que la propriété précédente (projection par des constantes de $\{-1, 0, 1, 2\}$) soit préservée.

Soit $i > 0$. On suppose que l'on a déjà construit $C\Sigma^{(i-1)}_n(\bar{a}, \bar{y})$ pour tout n, où la taille de $\bar{y}$ est n (dans la notation précédente, pour G_n on a regroupé $\bar{a}$ et $\bar{b}$ dans un seul $\bar{a}$, en effectuant juste un renommage ; on procèdera de même par la suite).

On pose

$$C\Pi^{(i)}_n(\bar{a}, \bar{c}, \bar{y}) = \prod_{k=0}^{n} \prod_{\bar{\epsilon}\in\{0,1\}^{n-k}} (c_k C\Sigma^{(i-1)}_n(\bar{a}, y_1, \dots, y_k, \epsilon_1, \dots, \epsilon_{n-k}) + 1 - c_k),$$

où $\bar{c}$ est un n-uplet de nouvelles variables. Le terme à l'intérieur du produit est $\mathrm{V\Sigma^{i-1}P}$ (*cf* section précédente), donc $(C\Pi^{(i)}_n)$ est dans $\mathrm{V\Pi^iP}$.

Si on a une suite $\mathrm{V\Pi^iP}$ dont les éléments sont de la forme

$$f_n(x_1, \dots, x_{p(n)}) = \prod_{\bar{\epsilon}} g_n(\bar{x}, \bar{\epsilon})$$

avec (g_n) dans $\mathrm{V\Sigma^{i-1}P}$, alors (g_n) est projection de $(C\Sigma^{(i-1)}_{t(n)})$ par hypothèse de récurrence : elle s'écrit sous la forme

$$g_n(x_1, \dots, x_{p(n)}, y_1, \dots, y_{q(n)}) = C\Sigma^{(i-1)}_{t(n)}(\bar{u}, v_1, \dots, v_{(t-p-q)(n)}, \bar{x}, \bar{y})$$

où les u_j et v_j prennent leurs valeurs dans $\{-1, 0, 1, 2\}$. Alors f_n s'écrit sous la forme

$$f_n(\bar{x}) = C\Pi^{(i)}_{t(n)}(\bar{u}, \bar{c}, v_1, \dots, v_{(t-p-q)(n)}, \bar{x}, \bar{y'}),$$

où $\bar{y'}$ prend des valeurs quelconques (cet uple est là pour combler la place qui reste), et $c_{t(n)-q(n)} = 1$, alors que les autres c_j sont nuls. Donc (f_n) est une projection de $(C\Pi^{(i)}_n)$, dont les uples prennent leurs valeurs dans $\{-1, 0, 1, 2\}$. On condensera l'écriture en $C\Pi^{(i)}_n(\bar{a}, \bar{y})$, c'est-à-dire que $\bar{c}$ est maintenant compris dans $\bar{a}$. On a ainsi montré que $(C\Pi^{(i)}_n)$ est $\mathrm{V\Pi^iP}$-complète.

De même, on définit les polynômes de la suite complète pour $\mathrm{V\Sigma^iP}$ par :

$$C\Sigma^{(i)}_n(\bar{a}, \bar{c}, \bar{y}) = \sum_{k=0}^{n} \sum_{\bar{\epsilon}\in\{0,1\}^{n-k}} c_k C\Pi^{(i-1)}_n(\bar{a}, y_1, \dots, y_k, \epsilon_1, \dots, \epsilon_{n-k}),$$

où $\bar{c}$ est un n-uplet de nouvelles variables. Le terme à l'intérieur de la somme est $\mathrm{V\Pi^{i-1}P}$ (*cf* section précédente), donc $(C\Sigma^{(i)}_n)$ est dans $\mathrm{V\Sigma^iP}$.

On montre de la même façon que $(C\Sigma^{(i)}_n)$ est $\mathrm{V\Sigma^iP}$-complète.

1.5 Séparation et effondrement

Maintenant que nous avons établi les propriétés de base des classes de notre hiérarchie, ainsi que l'existence de suites complètes pour chaque classe, nous sommes en mesure d'étudier des problèmes de séparation ou d'effondrement de la hiérarchie. Nous commençons par un lemme sur l'effondrement de la hiérarchie.

1.5.1 Lemme Si $\mathrm{V\Pi^2P} \subseteq \mathrm{V\Sigma P}$, alors la hiérarchie s'effondre sur $\mathrm{V\Sigma P}$.

Preuve Par récurrence, montrons que $\mathrm{V\Pi}^i\mathrm{P} \subseteq \mathrm{V\Sigma P}$ et $\mathrm{V\Sigma}^i\mathrm{P} \subseteq \mathrm{V\Sigma P}$ pour tout i.

Le résultat est clair pour $i = 0$.

Pour $i \geq 1$: $\mathrm{V\Sigma}^i\mathrm{P}$ consiste en des sommes de $\mathrm{V\Pi}^{i-1}\mathrm{P}$, qui par récurrence est inclus dans $\mathrm{V\Sigma P}$. Or la somme de fonctions $\mathrm{V\Sigma P}$ est encore $\mathrm{V\Sigma P}$, d'où le résultat. $\mathrm{V\Pi}^i\mathrm{P}$ consiste en des produits de $\mathrm{V\Sigma}^{i-1}\mathrm{P}$, donc par récurrence en des produits de $\mathrm{V\Sigma P}$. Ainsi, $\mathrm{V\Pi}^i\mathrm{P} \subseteq \mathrm{V\Pi}^2\mathrm{P} \subseteq \mathrm{V\Sigma P}$. □

1.5.2 Remarque De la même façon (mais nous ne nous servirons pas de ce résultat ultérieurement), si $\mathrm{V\Sigma P} = \mathrm{V\Pi P} = \mathrm{V\Sigma^0P}$ alors la hiérarchie s'effondre sur $\mathrm{V\Sigma^0P}$.

1.5.1 Propagation d'un effondrement

Nous poursuivons notre étude par l'effet de la séparation de deux classes pour les niveaux voisins.

1.5.1.1 Proposition Pour $i \geq 1$,

$$\mathrm{V\Sigma}^{i-1}\mathrm{P} \neq \mathrm{V\Sigma}^i\mathrm{P} \Rightarrow \mathrm{V\Sigma}^i\mathrm{P} \neq \mathrm{V\Pi}^i\mathrm{P}.$$

Preuve Supposons que nous ayons une suite de polynômes (P_n) complète pour $\mathrm{V\Sigma}^i\mathrm{P}$, dont tous les éléments soient irréductibles et aient un monôme dont le coefficient est 1. Dans ce cas, si P_n s'écrit sous la forme d'un gros produit, tous les termes du produit sont -1 ou 1, sauf l'un d'entre eux qui est P_n au signe près. Donc si $(P_n) \in \mathrm{V\Pi}^i\mathrm{P}$, alors $(P_n) \in \mathrm{V\Sigma}^{i-1}\mathrm{P}$. Ainsi, si on suppose $\mathrm{V\Sigma}^i\mathrm{P} = \mathrm{V\Pi}^i\mathrm{P}$, (P_n) serait dans $\mathrm{V\Pi}^i\mathrm{P}$, donc dans $\mathrm{V\Sigma}^{i-1}\mathrm{P}$. Puisque la suite (P_n) est complète, on aurait $\mathrm{V\Sigma}^{i-1}\mathrm{P} = \mathrm{V\Sigma}^i\mathrm{P}$, et on a montré la contraposée.

Il ne reste donc plus qu'à exhiber une suite de polynômes (P_n) $\mathrm{V\Sigma}^i\mathrm{P}$-complète, et dont chaque élément est irréductible et contient un monôme dont le coefficient est 1. Soit $(C_n(x))$ (où x est un uple de variables) une suite $\mathrm{V\Sigma}^i\mathrm{P}$-complète quelconque (on peut prendre par exemple la suite $(C\Sigma_n^{(i)})$ de la section précédente). Pour tout n, on définit $D_n(x,z) = C_n(x) + z$, où z est une nouvelle variable.

Nous prétendons que pour tout n, D_n est irréductible. En effet, étudions le cas de D_n pour un certain n quelconque. Si D_n s'écrit $D_n(x,z) = A(x,z)B(x,z)$, montrons que A ou B est constant. Puisque D_n est de degré 1 en z, on peut supposer que A est de degré 1 en z, et que z n'intervient pas dans B. Ainsi,

$$C_n(x) = D_n(x,0) = A(x,0)B(x).$$

Donc $B(x)$ divise $C_n(x)$, et $D_n(x,z)$ (par hypothèse), donc $B(x)$ divise z. Puisque z n'intervient pas dans B, on en déduit que B est constant. Ainsi, D_n est irréductible.

Bien entendu, $(D_n) \in \mathrm{V\Sigma}^i\mathrm{P}$. De plus, (D_n) est complète puisqu'il suffit de prendre $z = 0$ pour retrouver (C_n). On a donc trouvé notre suite complète de polynômes irréductibles, dont le monôme z a pour coefficient 1. □

De la même façon, nous avons le résultat suivant, qui a son importance en raison de la complétude de (HC_n) pour la classe VNP^0 (sans constante et de degré polynomialement borné), *cf* [15]. Le polynôme HC_n (le hamiltonien d'une matrice) est défini comme le permanent, mais la somme se fait seulement sur les permutations $\sigma \in \mathcal{S}_n$ qui sont des cycles de longueur n :

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

Le permanent n'est VNP^0-complet qu'en caractéristique différente de 2, c'est pourquoi nous préférons utiliser le hamiltonien, dont la complétude est vraie en toute caractéristique. Nous reviendrons sur ce polynôme au second chapitre.

1.5.1.2 Proposition

$$(HC_n) \notin \mathrm{V}\Sigma^0\mathrm{P} \Rightarrow (HC_n) \notin \mathrm{V}\Pi\mathrm{P}$$

Preuve La preuve repose comme précédemment sur l'irréductibilité du polynôme HC_n, que nous allons montrer, et sur le fait que le coefficient de chaque monôme de HC_n est 1.

Pour simplifier les notations, on fixe un entier n et on note $P = HC_n$. Supposons que $P = AB$ où A et B sont des polynômes. Si A contient la variable $x_{i,j}$ et B contient $x_{i,k}$ (où i, j et k sont quelconques), alors en écrivant A comme un polynôme en $x_{i,j}$ et B comme un polynôme en $x_{i,k}$, le produit contient au moins un terme contenant $x_{i,j}x_{i,k}$. Ce qui est impossible puisque $AB = P$ ne contient pas de tel terme. En raisonnant de même sur $x_{j,i}$ et $x_{k,i}$, on voit que dès que A contient $x_{i,j}$ il contient toutes les variables de la «ligne» $(x_{i,l})_{1\leq l\leq n}$ et toutes celles la «colonne» $(x_{l,j})_{1\leq l\leq n}$. Il contient donc toutes les variables de la «matrice» $(x_{l,m})$, et B est une constante. Ainsi, P est irréductible. □

1.5.2 Sur la complexité des fonctions coefficients

Nous nous attachons maintenant aux fonctions coefficient des polynômes des suites de notre hiérarchie, pour en déduire des conditions d'effondrement.

Fonctions coefficients de $\mathrm{V}\Sigma\mathrm{P}$

Pour un polynôme $p(n)$, on note $\bar{\epsilon} = (\bar{\epsilon}_1, \ldots, \bar{\epsilon}_{p(n)})$ un uple d'entiers naturels $\bar{\epsilon}_i$ codés en binaire. Ces entiers seront les puissances des indéterminées X_i.

Puisque le degré d'un polynôme calculé par un circuit de taille polynomiale est au plus simplement exponentiel, il existe un polynôme $q(n)$ tel que chaque entier $\bar{\epsilon}_i$ est au plus égal à $2^{q(n)}$, donc contient un nombre de chiffres binaires inférieur ou égal à $q(n)$. Ainsi, la longueur totale de $\bar{\epsilon}$ est au plus $p(n)q(n)$. Le résultat suivant remonte essentiellement à [21] : c'en est une simple généralisation à notre contexte (degré non polynomialement borné).

1.5.2.1 Proposition Supposons que la fonction $f : \{0,1\}^* \to \mathbf{Z}, (\bar{\epsilon}, n) \mapsto f(\bar{\epsilon}, n)$ soit dans GapP/poly, où n est codé en unaire. Alors la suite des polynômes

$$\phi_n(X_1, \ldots, X_{p(n)}) = \sum_{\bar{\epsilon}} f(\bar{\epsilon}, n) X_1^{\bar{\epsilon}_1} \cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}}$$

est dans $\mathrm{V}\Sigma\mathrm{P}$.

Preuve Par définition de GapP/poly, il existe une fonction g de GapP et une fonction de conseil $u(n)$ (ne dépendant que de l'entier n, et dont la taille est bornée par un polynôme), telle que $g(\bar{\epsilon}, n, u(n)) = f(\bar{\epsilon}, n)$. Par définition de GapP, il existe deux fonctions g_1 et g_2 de #P telles que $g = g_1 - g_2$.

Or pour $i \in \{1,2\}$, $g_i(\bar{\epsilon}, n, u(n)) = \mathrm{card}\{\bar{x} : \phi^{(i)}_{(\bar{\epsilon},n,u(n))}(\bar{x}) \text{ est vraie}\}$ pour une certaine formule booléenne $\phi^{(i)}_{(\bar{\epsilon},n,u(n))}$, calculable en temps polynomial à partir de $(\bar{\epsilon}, n, u(n))$ (complétude de #SAT pour les réductions parsimonieuses). $\phi^{(i)}_{(\bar{\epsilon},n,u(n))}(\bar{x})$ peut être aisément calculée par une suite de polynômes $(P_n^{(i)}(\bar{\epsilon}, \bar{x}))$ de $\mathrm{V}\Sigma^0\mathrm{P}$, qui prend les mêmes valeurs sur $\{0,1\}^*$. Soit $P_n = P_n^{(1)} - P_n^{(2)}$: cette suite de polynômes est aussi calculable en taille polynomiale.

Alors $g(\bar{\epsilon}, n, u(n)) = \sum_{\bar{x}} P_n(\bar{\epsilon}, n, \bar{x})$, et donc

$$\phi_n(X_1, \ldots, X_{p(n)}) = \sum_{\bar{\epsilon}} \sum_{\bar{x}} P_n(\bar{\epsilon}, n, \bar{x}) X_1^{\bar{\epsilon}_1} \cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}}.$$

Maintenant, l'élévation de chaque variable X_i à la puissance $\bar{\epsilon_i}$ donnée en binaire se fait en taille polynomiale en n (mais pas en degré borné), puisque chaque $\bar{\epsilon_i}$ a un nombre polynomial de chiffres. En effet, pour calculer Y^l, où les chiffres binaires de l sont notés l_j, on élève Y successivement au carré, et pour chaque résultat Y^{2^j} obtenu, on calcule $l_jY^{2^j}+1-l_j$. Enfin, on multiplie tous ces résultats, qui valent Y^{2^j} si l_j vaut 1, et 1 si $l_j=0$.

Ainsi, la suite à l'intérieur de la double somme est dans $\mathrm{V}\Sigma^0\mathrm{P}$, donc $(\phi_n)\in\mathrm{V}\Sigma\mathrm{P}$. □

Nous en déduisons que l'on peut aussi calculer la fonction coefficient chiffre par chiffre (notons toutefois que dans le corollaire suivant, f ne prend pas nécessairement ses valeurs dans $\{0,1\}$) :

1.5.2.2 Corollaire Supposons que la fonction $f:(\bar{\epsilon},i,n)\mapsto f(\bar{\epsilon},i,n)$ soit dans GapP/poly, où n est codé en unaire, et i représente un nombre codé en binaire de taille $p(n)$ pour un certain polynôme p. Alors la suite constituée des polynômes suivants est dans $\mathrm{V}\Sigma\mathrm{P}$:

$$\phi_n(X_1,\dots,X_{p(n)})=\sum_{\bar{\epsilon}}(\sum_{i=0}^{2^{p(n)-1}}f(\bar{\epsilon},i,n)2^i)X_1^{\bar{\epsilon}_1}\cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}}.$$

Preuve

$$\sum_{\bar{\epsilon}}(\sum_{i=0}^{2^{p(n)-1}}f(\bar{\epsilon},i,n)T^i)X_1^{\bar{\epsilon}_1}\cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}}=\sum_{\bar{\epsilon},i}f(\bar{\epsilon},i,n)T^iX_1^{\bar{\epsilon}_1}\cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}},$$

donc en appliquant la proposition précédente et en évaluant en $T=2$, on obtient le résultat. □

Notons qu'en degré polynomialement borné (ce qui, rappelons-le, n'est pas le cadre de notre étude), nous avons la réciproque, c'est-à-dire que la fonction coefficient est dans GapP/poly. On pourra se référer à [15] pour une étude plus complète des fonctions coefficient des classes VP et VNP, en degré borné ou non.

Fonctions coefficients de $\mathrm{V}\Pi\mathrm{P}$

Nous nous intéressons maintenant à calculer les coefficients des polynômes des suites de $\mathrm{V}\Pi\mathrm{P}$. Nous montrons qu'en caractéristique $p>0$, la fonction coefficient de tels polynômes est calculable en espace polynomial (la fonction appartient à la classe FPSPACE), de même pour $\mathrm{V}\Pi^2\mathrm{P}$. En caractéristique nulle, le nombre de chiffres peut être exponentiel, donc il faut calculer la fonction coefficient chiffre par chiffre. Ces considérations nous permettent de déduire un effondrement de la hiérarchie sur $\mathrm{V}\Sigma\mathrm{P}$ sous l'hypothèse FPSPACE $\subseteq$ GapP/poly.

Dans la suite, pour alléger les notations, on écrira $\bar{X}^{\bar{\epsilon}}$ pour $X_1^{\bar{\epsilon}_1}\cdots X_{p(n)}^{\bar{\epsilon}_{p(n)}}$, où $\bar{\epsilon}$ est un uple d'entiers écrits en binaire, comme précédemment. Nous séparons les cas où la caractéristique est nulle, puisque les coefficients peuvent avoir un nombre exponentiel de chiffres. Dans ce cas, nous calculons alors la fonction coefficient chiffre par chiffre.

1.5.2.3 Proposition

(i) Sur un corps de caractéristique $p>0$, supposons que la suite des polynômes

$$\phi_n(\bar{X})=\prod_{\bar{\eta}}g_n(\bar{\eta},\bar{X})=\sum_{\bar{\epsilon}}f(\bar{\epsilon},n)\bar{X}^{\bar{\epsilon}}$$

soit dans $\mathrm{V}\Pi\mathrm{P}$, c'est-à-dire $(g_n)\in\mathrm{V}\Sigma^0\mathrm{P}$. Alors $f\in$ FPSPACE.

(ii) Sur un corps de caractéristique nulle, supposons que la suite des polynômes

$$\phi_n(\bar{X})=\prod_{\bar{\eta}}g_n(\bar{\eta},\bar{X})=\sum_{\bar{\epsilon}}(\sum_{i=0}^{2^{p(n)}-1}f(\bar{\epsilon},i,n)2^i)\bar{X}^{\bar{\epsilon}},$$

où f est à valeurs dans $\{0,1\}$, soit dans $\mathrm{V}\Pi\mathrm{P}$ (c'est-à-dire $(g_n)\in\mathrm{V}\Sigma^0\mathrm{P}$). Alors $f\in$ FPSPACE.

Preuve

(i) Nous utilisons la caractérisation de FPSPACE en termes de circuits de profondeur polynomiale. Il suffit de montrer que l'on peut calculer chaque terme de $\phi_n(\bar{X})$ par un circuit booléen de profondeur polynomiale en n (et de taille éventuellement exponentielle).

On code chaque terme $\alpha\bar{X}^{\bar{\epsilon}}$ par l'uple $(\alpha, \bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$, où α est un entier relatif codé en binaire. Puisqu'on est en caractéristique positive, la taille de α sera bornée par une constante, et le degré du polynôme étant majoré par $2^{q(n)}$ pour un certain polynôme q, la taille de $\bar{\epsilon}_i$ est majorée par $q(n)$. Ainsi, la taille du codage d'un terme est polynomiale en n.

Au départ, on calcule les termes de $g_n(\bar{\eta}, \bar{X})$ en profondeur polynomiale (puisque $(g_n) \in \mathrm{V}\Sigma^0\mathrm{P}$), en parallèle pour chaque $\bar{\eta}$. Puis par récurrence, à l'étape i on calcule tous les termes de $\prod_{\bar{\eta}} g_n(\bar{\eta}, \bar{X})$, où la sommation concerne tous les $\bar{\eta}$ d'un intervalle de taille 2^i, en parallèle. Cela s'effectue en profondeur polynomiale, à partir des produits sur les deux demi-intervalles, calculés à l'étape de récurrence précédente : les produits de tous les termes deux à deux s'effectuent en une profondeur logarithmique (la multiplication des coefficients se fait en profondeur constante, et l'addition des puissances requiert une profondeur $\log(n^{O(1)})$), puis les additions des monômes égaux se fait en profondeur polynomiale puisqu'il y a au maximum un nombre simplement exponentiel de termes.

Puisqu'il y a un nombre polynomial d'étapes, la fonction f est bien dans FPSPACE.

(ii) Par le même raisonnement, on voit que l'on peut calculer le résultat chiffre par chiffre, tout en restant de profondeur polynomiale.

□

De même, on peut calculer la fonction coefficient des polynômes des suites de VΣP en espace polynomial. En composant avec le résultat précédent, on en déduit que la fonction coefficient des polynômes des suites de VΠ^2P est dans FPSPACE.

1.5.2.4 Corollaire Sur un corps de caractéristique quelconque, si FPSPACE $\subseteq$ GapP/poly, alors la hiérarchie s'effondre sur VΣP.

Preuve D'après les résultats précédents sur les fonctions coefficients de VΣP (proposition 1.5.2.1 et corollaire 1.5.2.2), l'hypothèse implique que VΠP $\subseteq$ VΣP. De même, VΠ^2P $\subseteq$ VΣP. Par le lemme 1.5.1, la hiérarchie s'effondre alors sur VΣP. □

L'hypothèse FPSPACE $\subseteq$ GapP/poly n'a pas une signification très intuitive. En annexe, nous essayons de mieux comprendre cette hypothèse grâce à des classes de langages plutôt que des classes de fonctions.

1.6 Circuit de séparation d'hyperplans

Dans cette partie, nous nous servons de notre hiérarchie pour construire un circuit de séparation d'hyperplans : grâce à des tests VΠP, on peut séparer des hyperplans affines sur K (un corps quelconque) par des circuits algébriques de taille $O(n \log N)$, où n est la dimension de l'espace, et N le nombre d'hyperplans. Ensuite, l'utilisation de vecteurs de coordonnées rationnelles de petite taille nous permettra de montrer un résultat de transfert concernant les versions non-uniformes des classes NP sur $(K, +, -, =)$ et P sur $(K, +, -, \times, =)$, que nous définissons ci-dessous.

Un langage L sur K est un sous-ensemble de $K^\infty = \cup_{n\geq 0} K^n$: un mot de longeur n est un uple de n éléments de K. Une machine BSS sur $(K, +, -, =)$ ([5], on pourra aussi se référer au livre [4]) est une machine de Turing qui manipule des éléments de K et est capable de les ajouter ou de les soustraire en une étape. La machine peut également tester l'égalité de deux éléments de K. Sur $(K, +, -, \times, =)$, elle peut en outre multiplier deux éléments en une étape. La classe $\mathrm{P}_{(K,+,-,\times,=)}$ (resp. $\mathrm{NP}_{(K,+,-,=)}$) est l'ensemble des langages L sur K reconnus par une machine BSS déterministe (resp. non déterministe) sur $(K, +, -, \times, =)$ (resp. $(K, +, -, =)$) en temps polynomial. La version non uniforme de $\mathrm{P}_{(K,+,-,\times,=)}$, notée $\mathbb{P}_{(K,+,-,\times,=)}$, est l'ensemble des langages L sur K reconnus par une famille de circuits algébriques de taille polynomiale, munis de portes de test pour l'égalité. On a $\mathrm{P}_{(K,+,-,\times,=)} \subseteq \mathbb{P}_{(K,+,-,\times,=)}$.

1.6.1 Problème

Soit K un corps quelconque. Un *hyperplan affine* de K^n est un ensemble de la forme $H = \{x \in K^n : \sum_{i=1}^n \alpha_i x_i = \beta\}$, où α_i et β sont des constantes fixées de K. Lorsque la constante β est nulle (*i.e.* lorsque l'hyperplan affine passe par 0), on parlera d'*hyperplan vectoriel*.

On considère une famille d'hyperplans affines (H_i), $i \in I$, de K^n. On définit la relation d'équivalence suivante sur K^n : $x \sim y$ si et seulement si $\forall i \in I$, $x \in H_i \Leftrightarrow y \in H_i$. Chaque classe d'équivalence pour cette relation est appelée *cellule* de la famille d'hyperplans affines. Dans ce qui suit, nous travaillerons sur des familles finies, ce qui implique que le nombre de cellules sera fini.

On suppose que pour chaque dimension n, on a une famille $\mathcal{H}_n$ d'hyperplans affines à séparer. Soit $(f_n(x_1, \ldots, x_{p(n)}))$ une suite de VΠP. Une *famille de circuits sur K^n avec tests (f_n)* est une suite de circuits algébriques (C_n), ayant n variables en entrée (en plus des constantes -1, 0 et 1), et munis de portes de tests : l'entrée de telles portes est un uple $(a_1, \ldots, a_{p(n)})$, et leur sortie est 0 si $f_n(a_1, \ldots, a_p(n)) = 0$, et 1 sinon. La sortie de tels circuits sur l'entrée $x \in K^n$ est un uple de valeurs de $\{0, 1\}$, noté $C_n(x)$. La famille de circuits *sépare* les familles $\mathcal{H}_n$ si pour tout n, pour tous $x, y \in K^n$, si x et y ne sont pas dans la même cellule de $\mathcal{H}_n$ alors $C_n(x) \neq C_n(y)$.

Pour tout $n \geq 0$, on appelle $(H_i^{(n)})$ les hyperplans de la famille $\mathcal{H}_n$, pour i de 1 à N_n. On note respectivement $f_i^{(n)}(x) = 0$ les équations les définissant, où les $f_i^{(n)}$ sont affines. On suppose que $N_n = 2^{n^{O(1)}}$, où n est la dimension de l'espace. On suppose que la suite des fonctions $\phi_n : (i, x) \mapsto f_i^{(n)}(x)$, où i est codé en binaire, est dans $\mathrm{V\Sigma^0P}$ (c'est-à-dire calculable par une suite de circuits de taille polynomiale, sans constante arbitraire de K). Nous cherchons une suite de circuits qui sépare nos familles d'hyperplans, et qui soit de taille petite. Si le nombre de cellules est c, une borne inférieure évidente sur la profondeur du circuit est $\lceil \log c \rceil$, puisque si l'on déplie le circuit en un arbre binaire, il doit avoir au moins c feuilles.

Dans la première partie, nous montrons comment on peut se ramener aux cas d'hyperplans vectoriels, en travaillant dans une dimension supérieure. Nous construisons ensuite une famille de circuits séparant la famille des (H_i) de taille $O(n \log N)$, où n est la dimension de l'espace, et N le nombre d'hyperplans de la famille.

On se place maintenant dans une dimension n fixée, mais l'hypothèse que $(\phi_n) \in \mathrm{V\Sigma^0P}$ fournit l'uniformité de la construction pour tout n.

1.6.2 Se ramener à des hyperplans vectoriels

Supposons que nous sachions résoudre le problème dans le cas d'hyperplans vectoriels, c'est-à-dire que pour toute famille d'hyperplans vectoriels vérifiant la condition $(\phi_n) \in \mathrm{V\Sigma^0P}$, nous sachions construire un circuit la séparant de taille $O(n \log N)$.

Nous allons construire un circuit séparant notre famille d'hyperplans affines (H_i). Nous définissons d'abord une famille d'hyperplans vectoriels (H'_i) de K^{n+1} ainsi : si H_i est défini par $\sum_{i=1}^n \alpha_i x_i = \beta$, alors H'_i est défini par $\sum_{i=1}^n \alpha_i x_i - \beta x_{n+1} = 0$. La propriété suivante est alors vérifiée : $(x_1, \ldots, x_n) \in H_i$ si et seulement si $(x_1, \ldots, x_n, 1) \in H'_i$. On appelle C' le circuit que l'on sait construire pour séparer la famille des (H'_i).

On appelle C le circuit sur K^n obtenu en remplaçant dans C' la variable x_{n+1} par 1, en particulier chaque test $f_n(x_1, \ldots, x_{n+1}) = 0$ est remplacé par $f_n(x_1, \ldots, x_n, 1) = 0$. Soient x et y deux points de K^n. Si x et y ne sont pas dans la même cellule de la famille des H_i, par exemple il existe i_0 tel que $x \in H_{i_0}$ et $y \notin H_{i_0}$, alors $x' := (x_1, \ldots, x_n, 1) \in H'_{i_0}$ et $y' := (y_1, \ldots, y_n, 1) \notin H'_{i_0}$, donc $C'(x') \neq C'(y')$, ce qui implique que $C(x) \neq C(y)$, par définition de C.

Ainsi, C fournit une solution à notre problème. Dans la suite, on supposera donc que l'on a une famille d'hyperplans vectoriels (et pas seulement affines).

1.6.3 Le circuit dans le cas d'hyperplans vectoriels

On se donne donc une famille d'hyperplans vectoriels (H_i) de K^n, i entier entre 1 et N, définis respectivement par $f_i(x) = 0$, où les f_i sont linéaires.

On maintient un espace E qui est le plus petit espace dont on est sûr qu'il contient x. C'est l'intersection de tous les hyperplans qui contiennent x que l'on a trouvés jusqu'à présent. Au départ, $E = K^n$. À chaque étape, on trouve un hyperplan H contenant notre point x, mais dont la trace n'est pas triviale sur E (*i.e.* $E \cap H \neq E$). Pour cela, on effectue une recherche dichotomique parmi tous les hyperplans dont la trace sur E n'est pas triviale, afin de trouver le premier contenant x. Ainsi, à chaque étape la dimension diminue de E de 1, donc on localise x en au plus n étapes.

Le circuit de séparation

Si α est un vecteur de K^n, on notera H_α l'hyperplan défini par $\sum \alpha_j x_j = 0$. On aura besoin du résultat technique suivant.

1.6.3.1 Lemme Soient $\alpha^{(1)}, \ldots, \alpha^{(k)}$ des vecteurs de K^n linéairement indépendants. Soit β un autre vecteur. Alors la famille $(\alpha^{(1)}, \ldots, \alpha^{(k)}, \beta)$ est liée si et seulement si $H_{\alpha^{(1)}} \cap \ldots \cap H_{\alpha^{(k)}} \subseteq H_\beta$.

Preuve Supposons que la famille $(\alpha^{(1)}, \ldots, \alpha^{(k)}, \beta)$ soit liée. Puisque les vecteurs $\alpha^{(1)}, \ldots, \alpha^{(k)}$ sont linéairement indépendants, il existe $\gamma_1, \ldots, \gamma_k$ des scalaires de K tels que $\beta = \sum_{i=1}^k \gamma_i \alpha^{(i)}$. Ainsi, si l'on prend x dans $H_{\alpha^{(1)}} \cap \ldots \cap H_{\alpha^{(k)}}$, alors $\sum_{j=1}^n \alpha_j^{(i)} x_j = 0$ pour tout i, donc

$$\sum_{j=1}^n \beta_j x_j = \sum_{j=1}^n \sum_{i=1}^k \gamma_i \alpha_j^{(i)} x_j = \sum_{i=1}^k \gamma_i \sum_{j=1}^n \alpha_j^{(i)} x_j = 0,$$

d'où l'inclusion.

Réciproquement, soit A la matrice

$$A = \begin{pmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(k)} \\ \beta \end{pmatrix}.$$

Si la famille $(\alpha^{(1)}, \ldots, \alpha^{(k)}, \beta)$ est libre, alors A est de rang $k+1$, donc le système

$$Ax = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

a une solution x, vérifiant pour tout i, $\sum_j \alpha_j^{(i)} x_j = 0$, et $\sum_j \beta_j x_j = 1$. Ainsi, $x \in (\cap_i H_{\alpha^{(i)}}) \backslash H_\beta$, donc on a montré la réciproque. □

Comme mentionné précédemment, on maintient un espace E localisant x le plus précisément possible avec les informations que nous possédons. Au départ, $E = K^n$, et après l'étape k, $E = H_{\alpha^{(1)}} \cap \ldots \cap H_{\alpha^{(k)}}$, où les $\alpha^{(i)}$ sont linéairement indépendants.

Décrivons maintenant l'étape $k+1$. On effectue une recherche dichotomique parmi tous les hyperplans H dont la trace sur E est non triviale (c'est-à-dire $E \not\subseteq H$), pour trouver le premier contenant x. Pour cela, si on note I l'ensemble des hyperplans de trace non triviale sur E, on effectue $O(\log N)$ produits de la forme

$$\prod_{i \in I, i < j} \phi_n(i, x),$$

que l'on décrit en détail à la fin de cette section. En effet, ces produits nous permettent de déterminer s'il existe un hyperplan $< j$, de trace non triviale, et contenant x : si oui, on diminue l'intervalle (on fait le produit sur $i < (j + j_{\min})/2$, où $j_{\min}$ est la borne du plus grand intervalle visité jusqu'à présent et contenu dans l'intervalle en cours, avec au départ $j_{\min} = 0$) ; sinon, on agrandit l'intervalle (on fait le produit sur $i < (j + j')/2$, où j' était la borne de l'intervalle précédent). Lorsque l'on a trouvé $H = H_\alpha$ qui convient, on est sûr que α est linéairement indépendant des $\alpha^{(i)}$, d'après le lemme 1.6.3.1.

On renouvelle alors E : maintenant, $E = H_{\alpha^{(1)}} \cap \ldots \cap H_{\alpha^{(k)}} \cap H_\alpha$, et on a fini l'étape $k+1$.

Si en revanche, on ne trouve aucun hyperplan H convenable, alors on arrête la construction, et la sortie du circuit est le uple constitué du numéro de chaque hyperplan trouvé (en binaire).

Il reste à montrer que le circuit ainsi construit sépare notre famille d'hyperplans. Soient x et y deux points dans des cellules différentes : par exemple $x \in H$ et $y \notin H$, pour un certain H. Supposons que $C(x) = C(y)$, et que le processus se soit terminé à l'étape k. Alors il existe une suite d'hyperplans H_{i_j}, pour $j = 1$ à k, contenant x et y. H ne fait pas partie de ces hyperplans, puisque y n'est pas dans H. Pour la même raison, la trace de H n'est pas triviale sur $\cap_j H_{i_j}$. Donc le processus ne devrait pas s'arrêter pour x, car l'hyperplan H convient : contradiction.

Produits $\mathrm{V\Pi P}$ **pour la recherche dichotomique**

Il reste à montrer que l'on peut faire dans $\mathrm{V\Pi P}$ des produits pour la recherche dichotomique de la forme

$$(j,k,x) \mapsto \prod_{i \in I, i<j} \phi_n(i,x),$$

où j est un entier binaire, et k code en binaire les numéros de tous les hyperplans trouvés jusqu'à présent. j se calcule facilement par le circuit, puisque c'est la borne de la recherche dichotomique. Pour I, la proposition 1.3.2.2 nous permet de conclure : il suffit pour cela de montrer que décider si un hyperplan est de trace non triviale sur E (que l'on connaît grâce à k) est dans P/poly. D'après le lemme 1.6.3.1, c'est un simple calcul de rang, puisque la trace de H_α est triviale sur $H_{\alpha^{(1)}} \cap \ldots \cap H_{\alpha^{(k)}}$ si et seulement si la famille $(\alpha^{(1)}, \ldots, \alpha^{(k)}, \alpha)$ est liée (en effet, on a pris soin lors de la construction de vérifier que les $\alpha^{(i)}$ étaient toujours indépendants). Cela se fait dans P/poly, par exemple par élimination de Gauss.

1.6.4 Conséquences de notre construction

Soit K un corps de caractéristique nulle. Pour tout langage L dans $\mathrm{NP}_{(K,+,-,=)}$, nous déduisons de notre construction une famille de circuits avec tests $\mathrm{V\Pi P}$ qui reconnaît L.

Nous montrons d'abord des lemmes sur l'existence d'un vecteur à coordonnées rationnelles de petite taille dans chaque cellule d'un arrangement d'hyperplans. Un vecteur *à coordonnées rationnelles de taille* $\leq k$ est un vecteur dont chaque coordonnée est un rationnel, de numérateur et dénominateur majorés par 2^k. Commençons par calculer la taille de la somme et du produit de rationnels.

1.6.4.1 Lemme Soient α et β deux rationnels de taille t et t' respectivement. Alors

- $\alpha\beta$ est de taille $\leq t + t'$;
- $\alpha + \beta$ est de taille $\leq t + t' + 1$.

En particulier, si A est une matrice de taille $n \times m$ dont tous les coefficients sont des rationnels de taille $\leq t$, et si x est un vecteur de taille n dont tous les coefficients sont des rationnels de taille $\leq t'$, alors Ax est un vecteur de taille m dont tous les coefficients sont des rationnels de taille $\leq n(t+t') + n - 1$.

Preuve Pour les deux premiers points, on utilise simplement les égalités

$$\frac{p}{q}\frac{p'}{q'} = \frac{pp'}{qq'} \text{ et } \frac{p}{q} + \frac{p'}{q'} = \frac{pq' + p'q}{qq'}.$$

Pour les produits matrices/vecteurs, chaque coefficient de Ax est la somme de n produits d'un coefficient de A par un coefficient de x. □

On dit que n hyperplans sont *indépendants* si les n vecteurs les définissant forment une famille libre. On dit qu'une cellule C d'un arrangement d'hyperplans de K^n est *définie par les m hyperplans indépendants* $H_1, \ldots, H_m$ (avec $m \leq n$) si $X = \{H_1, \ldots, H_m\}$ est un ensemble maximal pour l'inclusion parmi les ensembles dont les éléments sont des hyperplans indépendants et vérifiant $C \subseteq \cap X$. En d'autres termes, l'intersection des m hyperplans de X est l'espace défini par les équations $f_i(x) = 0$ de la cellule C (on ne prend pas en compte les inéquations $g_i(x) \neq 0$). Notons que cela décrit de manière unique notre cellule.

1.6.4.2 Lemme Soit $\mathcal{H}$ une famille d'hyperplans de K^n dont les coefficients sont des entiers majorés en valeur absolue par k. Alors le vecteur x de coordonnées $x_i = (k+1)^i$ (pour i de 1 à n) n'est dans aucun hyperplan de la famille $\mathcal{H}$.

Preuve Soit $f(x) = \sum_{i=1}^n \alpha_i x_i$ l'équation d'un hyperplan de la famille $\mathcal{H}$. On écrit $f^+(x) = \sum_{i=1}^n \alpha_i^+ x_i$ et $f^-(x) = \sum_{i=1}^n \alpha_i^- x_i$, où $\alpha_i^+ = \max(0, \alpha_i)$ et $\alpha_i^- = \max(0, -\alpha_i)$. Alors $f(x) = 0$ équivaut à $f^+(x) = f^-(x)$, c'est-à-dire (en utilisant l'expression de x) $\sum_{i=1}^n \alpha_i^+(k+1)^i = \sum_{i=1}^n \alpha_i^-(k+1)^i$. Mais puisque les coefficients α_i^+ et α_i^- sont tous positifs et bornés par k, par unicité de l'écriture en base $(k+1)$ on en déduit que $f(x) = 0$ équivaut à $\forall i, \alpha_i = 0$, donc dans ce cas, H ne serait pas un hyperplan. Ainsi, x n'est dans aucun hyperplan de $\mathcal{H}$. □

1.6.4.3 Lemme Soit $\mathcal{H}$ une famille d'hyperplans de K^n dont les coefficients des équations sont des entiers majorés en valeur absolue par k. Supposons qu'un circuit booléen de taille t nous fournisse l'équation du i-ème hyperplan de $\mathcal{H}$ sur l'entrée binaire i. Alors il existe un circuit de taille polynomiale en n, t et $\log k$, qui sur l'entrée constituée des numéros de m hyperplans indépendants ($m \leq n$), retourne un vecteur x vérifiant :
- x est dans la cellule définie par les m hyperplans ;
- x est à coordonnées rationnelles de taille polynomiale en n et $\log k$.

Preuve On appelle E l'intersection des m hyperplans $H_1, \ldots, H_m$ donnés en entrée, et on appelle f_i l'équation de H_i. Par pivot de Gauss, on peut se ramener à un système triangulaire équivalent $f_i'(x) = 0$ $(1 \leq i \leq m)$, où l'on peut supposer que le coefficient de x_i dans f_i' est non nul. Les coefficients sont des rationnels de taille polynomiale en ceux des f_i.

On va se ramener au cas où $E = K^{n-m}$. On considère pour cela l'application

$$\begin{array}{lccc} \phi : & K^{n-m} & \to & K^n \\ & (x_{m+1}, \ldots, x_n) & \mapsto & (x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) \end{array}$$

où les $x_1, \ldots, x_m$ sont obtenus à partir des $x_{m+1}, \ldots, x_n$ grâce aux équations $f_i'(x) = 0$. ϕ est linéaire, et injective puisque l'on a m équations indépendantes pour trouver m points. L'image de ϕ est E, ce qui nous fournit une bijection entre espaces de même dimension. De plus, pour calculer les coefficients de ϕ dans la base canonique, il s'agit de résoudre le système des m équations $f_i'(x) = 0$, donc d'inverser la matrice carrée M de taille m dont les coefficients sont de taille polynomiale en ceux des f_i, par hypothèse majorés en valeur absolue par k. Soit k' un majorant de taille polynomiale des numérateurs et dénominateurs des coefficients des f_i'. En se rappelant que l'inverse d'une matrice est donné par le produit de l'inverse du déterminant et de la transposée de la matrice des cofacteurs, on remarque que chaque nouveau coefficient est un rationnel dont le numérateur et le dénominateur est majoré en valeur absolue par $(k'\sqrt{n})^n$ (on peut utiliser l'inégalité de Hadamard pour majorer le déterminant de M). Ainsi, les coefficients de ϕ sont des rationnels de taille $\leq n(\log k' + 1/2 \log n)$, donc si l'on applique ϕ à un vecteur de coordonnées rationnelles de taille $\leq r$, on obtient un vecteur de coordonnées rationnelles de taille $\leq n^2(\log k' + 1/2 \log n) + nr + n - 1$ (*cf* lemme 1.6.4.1), c'est-à-dire polynomiale en n, $\log k$ et r.

Soit H un hyperplan défini par une équation $g(x) = 0$ à coefficients entiers majorés en valeur absolue par k. Alors par substitution des variables $x_1, \ldots, x_m$, $\phi^{-1}(E \cap H)$ (dans K^{n-m}) est défini par des équations dont les coefficients sont de taille polynomiale en $\log k$ et n. Si on appelle b un entier de taille polynomiale majorant tous ces coefficients (que l'on peut calculer explicitement en temps polynomial grâce à l'expression de ϕ), par le lemme précédent, le vecteur $x \in K^{n-m}$ défini par $x_i = (b+1)^i$ (pour $1 \leq i \leq n-m$) n'est dans aucun hyperplan de K^{n-m} dont les coefficients sont des entiers majorés en valeur absolue par b. Ainsi, $\phi(x)$ est dans notre cellule C, puisqu'il est dans E mais dans aucun autre hyperplan de $\mathcal{H}$ de trace non triviale sur E, et ses coordonnées sont des rationnels de taille polynomiale en n et $\log k$.

Cela nous fournit l'existence du point cherché, et la procédure suivante pour le trouver.
- Par m appels au circuit donnant les équations des hyperplans, on connaît les m équations définissant notre cellule.
- Par pivot de Gauss, on se ramène à un système triangulaire $f_i'(x) = 0$ (tel que le coefficient de x_i dans la i-ème équation soit non nul).
- On calcule les coefficients de ϕ.

– Trouver un entier b de taille polynomiale majorant les coefficients des $\phi^{-1}(E \cap H)$.
– On définit x par $x_i = (b+1)^i$ pour $i = 1$ à m.
– On calcule $\phi(x)$.

Le circuit effectuant ces opérations a une taille polynomiale en n, t et $\log k$. □

Nous présentons maintenant le résultat principal de cette section.

1.6.4.4 Proposition On se place sur un corps K de caractéristique nulle. Pour tout langage $L \in \mathrm{NP}_{(K,+,-,=)}$, il existe une suite de circuits de taille polynomiale avec tests dans $\mathrm{V\Pi P}$ qui décide L.

Preuve Soit un langage $L \in \mathrm{NP}_{(K,+,-,=)}$. On appelle M la machine BSS non déterministe reconnaissant L en temps polynomial. Par le résultat de [12], dans $\mathrm{NP}_{(K,+,-,=)}$ on peut quantifier uniquement sur des uples booléens : il existe un langage $A \in \mathrm{P}_{(K,+,-,=)}$ et un polynôme $p(n)$ tels que

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

Soit $x = (x_1, \ldots, x_n)$ l'entrée de M. On considère les constantes de K utilisées par M (qui sont en nombre fini constant) comme de nouvelles entrées (*i.e.* $x_{n+1}, \ldots, x_{n+m}$). M ne fait que des tests de la forme $\sum_j \alpha_j x_j + \delta = 0$ (puisque l'on ne dispose que de l'addition et de l'égalité), où les coefficients α_j et δ sont des entiers dans $\{-2^{t(n)}, \ldots, 2^{t(n)}\}$ pour un certain polynôme $t(n)$ (on peut prendre pour $t(n)$ le temps mis par M pour reconnaître L). Ainsi, $L \cap K^n$ est la réunion de cellules de l'arrangement d'hyperplans $\mathcal{H}_n$ contenant tous les hyperplans de K^n dont les coefficients sont des entiers dans $\{-2^{t(n)}, \ldots, 2^{t(n)}\}$.

Il existe bien sûr un circuit de taille polynomiale en n qui sur l'entrée binaire i donne l'équation du i-ème hyperplan de $\mathcal{H}_n$. Remarquons de plus que la famille d'hyperplans $\mathcal{H}_n$ nous permet même de construire un circuit booléen de taille polynomiale qui, sur l'entrée (q,l) (où q est un point à coordonnées rationnelles donné en binaire, et l le numéro d'un hyperplan), renvoie 0 si q appartient à l'hyperplan l, et 1 sinon (ou inversement).

Puisque cette famille d'hyperplans vérifie les conditions de notre construction précédente (et même plus), on a une suite de circuits (C_n) avec tests dans $\mathrm{V\Pi P}$, qui sépare $\mathcal{H}_n$. Puisque $L \cap K^n$ est l'union de cellules, il suffit maintenant sur l'entrée x de déterminer si la cellule D de x est dans L ou non.

Puisque le circuit de séparation précédent nous donne en sortie le numéro des m hyperplans indépendants définissant D, on peut appliquer le lemme 1.6.4.3 pour trouver un vecteur q à coordonnées rationnelles de taille polynomiale en n et qui est dans la cellule D. Il suffit donc de déterminer s'il existe un uple $y \in \{0,1\}^{p(|x|)}$ tel que $(q,y) \in A$. Puisque q et y sont codés en binaires et q est un vecteur de coordonnées rationnelles de petite taille, on peut simuler le fonctionnement d'un circuit algébrique reconnaissant A par un circuit booléen B de taille polynomiale. B retourne 0 si (q,y) est dans A, et 1 sinon.

Par un gros produit sur tous les uples y, on peut ainsi tester s'il existe $y \in \{0,1\}^{p(|x|)}$ vérifiant $(q,y) \in A$, et ainsi résoudre le problème L. □

On en déduit le résultat de transfert suivant.

1.6.4.5 Corollaire Soit K un corps de caractéristique nulle. Si $\mathrm{V\Pi P} \subseteq \mathrm{V\Sigma^0 P}$ sur K, alors $\mathrm{NP}_{(K,+,-,=)} \subseteq \mathbb{P}_{(K,+,-,\times,=)}$.

1.6.4.6 Remarque L'hypothèse $\mathrm{V\Pi P} \subseteq \mathrm{V\Sigma^0 P}$ semble peu probable, car

– elle implique que la suite définie par

$$f_n(x) = \prod_{i=0}^{2^n-1}(x-i)$$

est dans $\mathrm{V\Sigma^0 P}$. On en déduit que si $\mathrm{V\Pi P} \subseteq \mathrm{V\Sigma^0 P}$, alors la τ-conjecture ([3]) est fausse.

– De même, si $\mathrm{V\Pi P} \subseteq \mathrm{V\Sigma^0 P}$, alors on calcule dans $\mathrm{V\Sigma^0 P}$ la suite $(k!)$ pour k exponentiel en n, ce qui implique que la factorielle est facile à calculer.

Chapitre 2

Manipulation de polynômes donnés par des circuits

2.1 Introduction

Dans ce chapitre, nous nous intéressons cette fois à la complexité booléenne de manipuler des polynômes donnés par des circuits. Un circuit algébrique est en effet facilement codable en binaire, puisqu'il suffit de décrire la structure du circuit (les arêtes) et le type des portes (addition, multiplication).

Cette manière de coder un polynôme peut être beacoup plus économique que de décrire chaque monôme : par exemple, le polynôme

$$(x+y)^{2^n}$$

est défini par un circuit de taille linéaire en n (par élévations au carré successives), mais il contient un nombre exponentiel de monômes. Toutefois, pour que ce codage soit utilisable, il faut être capable de manipuler de tels circuits... et ce n'est pas toujours facile !

Plusieurs travaux dans ce sens ont déjà été effectués : citons par exemple le calcul du PGCD de deux polynômes donnés par des circuits, dans [10]. Nous nous intéressons ici à deux problèmes naturels, CMN (Coefficient de Monôme Nul), et DEG (Degré), que nous décrivons ci-dessous. Nous nous plaçons dans le cadre des circuits algébriques sans constante, sur un corps quelconque.

Lorsque nous avons un circuit, on peut s'intéresser à son degré formel complet (terminologie reprise de [15]), qui intuitivement serait le degré du polynôme calculé si toutes les constantes (-1, 0 et 1) étaient remplacées par des variables. Dans ce cas, il n'y aurait aucune annulation (qui apparaîtrait par exemple lorsque l'on ajoute deux quantités opposées), puisque l'on n'a pas de porte de soustraction dans nos circuits. Formellement, on obtient la définition suivante.

2.1.1 Définition Le *degré formel complet* d'une porte d'un circuit algébrique est défini par récurrence :

- le degré formel complet d'une constante ou d'une variable est 1 ;
- le degré formel complet d'un porte $+$ est le max des degrés formels complets de ses entrées ;
- le degré formel complet d'un porte $\times$ est la somme des degrés formels complets de ses entrées.

Le degré formel complet d'un circuit est le degré formel complet de sa porte de sortie. On notera $\deg_{\mathrm{f}}(C)$ le degré formel complet du circuit C.

Il est facile de calculer le degré formel complet d'un circuit, par induction. Mais il semble beaucoup plus difficile de trouver le degré du polynôme que calcule ce circuit. En effet, comme mentionné précédemment, des annulations de monômes dominants peuvent avoir lieu, par exemple. On notera $\deg(C)$ le degré du polynôme calculé par le circuit C. Le problème DEG est le suivant :

$$\mathrm{DEG} = \{(C,d) : \deg(C) \leq d\}.$$

Le second problème auquel nous nous intéressons est celui du calcul du coefficient d'un monôme dans un circuit donné. Pour obtenir un problème de décision, sur l'entrée (C,m) nous nous demandons plus exactement si le coefficient du monôme m dans le polynôme calulé par C est nul. La notation $\mathrm{coef}_C(m)$ désignera ce coefficient. Le problème CMN est le suivant :

$$\mathrm{CMN} = \{(C,m) : \mathrm{coef}_C(m) = 0\}.$$

Rappelons que nous notons $|C|$ la taille du circuit C. Les cas où le degré formel complet du circuit d'entrée est grand (*i.e.* superpolynomial en $|C|$) posent en général problème. C'est pourquoi nous étudions dans un premier temps les versions «bornées» de nos problèmes : nous nous fixons un polynôme $p(n)$, et nous intéressons à

$$\mathrm{DEG_b} = \{(C,d) : \deg_{\mathrm{f}}(C) \leq p(|C|) \text{ et } \deg(C) \leq d\}$$

$$\mathrm{CMN_b} = \{(C,m) : \deg_{\mathrm{f}}(C) \leq p(|C|) \text{ et } \mathrm{coef}_C(m) = 0\}.$$

Notre étude dépend alors du polynôme $p(n)$, mais maintenant on sait que le degré formel complet du circuit est polynomial. Si l'on veut éviter de formuler nos problèmes à l'aide d'un polynôme $p(n)$, on peut utiliser par exemple la notion de *circuit multiplicativement disjoint* définie dans [15] : un circuit multiplicativement disjoint vérifie $\deg_{\mathrm{f}}(C) \leq |C|$, et tout circuit peut être transformé en un circuit multiplicativement disjoint en temps polynomial en $\deg_{\mathrm{f}}(C)$. On peut ainsi reformuler nos deux problèmes en termes de circuits multiplicativement disjoints : nos problèmes prennent alors en entrée (C,d) ou (C,m), avec C multiplicativement disjoint, et nous n'avons plus besoin du test $\deg_{\mathrm{f}}(C) \leq p(|C|)$. Les versions bornées de nos problèmes désignerons indifféremment des entrées de degré formel complet majoré par $p(|C|)$, ou des entrées multiplicativement disjointes.

Nous étudions dans un premier temps le calcul du degré dans la version bornée. Ensuite, nous nous intéressons à CMN, version bornée ou non.

2.2 Calcul du degré d'un polynôme dans le cas borné

Dans cette partie, nous obtenons un algorithme Co-RP pour le problème $\mathrm{DEG_b}$.

2.2.1 Calcul des composantes homogènes

On s'occupe d'abord du circuit C afin de séparer toutes les composantes homogènes (qui sont en nombre polynomial, d'après l'hypothèse sur le degré formel complet du circuit). [6, prop. 5.28] et [15, lemme 2] utilisent déjà cette technique pour «tronquer» un polynôme. On remplace chaque porte α du circuit par $p(|C|)+1$ portes, numérotées $(\alpha_0,\ldots,\alpha_{p(|C|)})$. La porte α_i calculera la composante homogène de degré i de la porte α.

Pour les portes d'entrée de notre circuit C : si la porte α est une variable X, alors les $p(|C|)+1$ nouvelles portes sont $(0,X,0,\ldots,0)$. De même, si la porte α est une constante a, alors les $p(|C|)+1$ nouvelles portes sont $(a,0,\ldots,0)$.

On construit alors inductivement le nouveau circuit comme suit (on a déjà traité le cas des entrées). Supposons que l'on en soit à la porte α, dont les deux entrées sont les portes β et γ. Deux cas se présentent.

- α est une porte $+$. Alors pour tout i, $\alpha_i = \beta_i + \gamma_i$.
- α est une porte $\times$. Alors pour tout i,

$$\alpha_i = \sum_{j+k=i} \beta_j \gamma_k,$$

  ce qui se calcule grâce à $i+1$ portes $\times$ supplémentaires, pour les produits, et à i portes d'addition. On ajoute donc $2i$ portes (sans compter α_i) pour calculer α_i. Ainsi, en tout, on a ajouté $p(|C|)(p(|C|)+1)$ portes.

Le nouveau circuit a ainsi une taille $\leq |C|(p(|C|) + p(|C|)(p(|C|)+1))$, qui reste polynomiale.

Par récurrence, on voit que la porte α_i calcule la composante homogène de degré i de la porte α. La procédure indiquée ci-dessus pour construire le nouveau circuit est clairement polynomiale. Ainsi, on dispose d'un nouveau circuit calculant séparément chaque composante homogène de notre circuit C de départ. Notons f_i la composante homogène de degré i (pour $0 \leq i \leq p(|C|)$) du polynôme f calculé par le circuit C.

Pour tester si $\deg(C) \leq d$, il suffit alors de tester si pour tout $i > d$, $f_i = 0$. Il y a donc un nombre polynomial de test de nullité d'un polynôme à effectuer. Ainsi, pour prouver que DEG $\in$ Co-RP, il suffit de montrer que tester la nullité d'un polynôme donné par un circuit se fait dans Co-RP.

2.2.2 Tester la nullité d'un polynôme

C'est un problème classique, qui est traité dans [17] par exemple. Pour tester la nullité d'un polynôme $f(x_1, \ldots, x_n)$, on choisit des n-uples $(a_1, \ldots, a_n)$ aléatoirement, parmi un ensemble «assez grand», et on calcule $f(a_1, \ldots, a_n)$. Si le résultat n'est pas nul, on est sûr que f n'est pas nul ; sinon, on réitère. Si après un certain nombre d'itérations, on n'a trouvé que des zéros de f, alors f est nul avec forte probabilité : formellement, on obtient un algorithme Co-RP.

Lorsque les résultats des calculs risquent de devenir trop grands, à cause d'un degré de polynôme exponentiel, par exemple, il est nécessaire d'effectuer les opérations arithmétiques modulo un nombre premier aléatoire, pour que l'algorithme reste polynomial. Cependant, puisque dans notre cas, le degré formel complet de C est majoré par $p(|C|)$, on peut effectuer les calculs directement : on est sûr que chacun des résultats intermédiaires s'écrira avec un nombre polynomial de bits.

Le seul point à résoudre est comment connaître un ensemble «assez grand» dans notre corps (puisque l'on souhaite choisir des points parmi un ensemble assez grand). En effet, dans le cas d'un corps de caractéristique nulle, cela ne pose pas de problème, mais lorsque l'on a un corps de caractéristique $p > 0$, puisque l'on ne dispose pas de constante arbitraire, on ne peut *a priori* accéder qu'au corps premier $\mathbf{F}_p$, donc à seulement p éléments. Ce qui n'est pas suffisant. Dans l'article [17], on a besoin d'un ensemble ayant au moins $2\deg(f)$ éléments pour avoir une probabilité d'erreur $\leq 1/2$.

Pour palier à cela, on travaille dans une extension de $\mathbf{F}_p$, définie par

$$\mathbf{F}_{p^q} \simeq \mathbf{F}_p[X]/(Q),$$

où Q est un polynôme irréductible de degré q de $\mathbf{F}_p$. Le corps $\mathbf{F}_{p^q}$ ayant p^q éléments, il suffit de prendre q assez grand pour que $p^q \geq 2\deg(f)$ (c'est-à-dire q de l'ordre de $\log \deg_{\mathrm{f}}(C)$).

Dans [11, prop. II.1.9], il est expliqué comment effectuer les calculs efficacement dans ce surcorps de $\mathbf{F}_p$: il s'agit d'additions et de multiplications de polynômes, suivies d'une division euclidienne par Q. Il ne reste donc plus qu'à trouver un polynôme irréductible de degré q. Pour cela, nous utilisons le corollaire de [11, prop. II.1.8] : pour q premier (ce que nous pouvons supposer sans perte de généralité), la proportion de polynômes unitaires irréductibles de degré q dans $\mathbf{F}_p$ est $\geq 1/(2q)$. Ainsi, en choisissant aléatoirement $O(q)$ polynômes unitaires, la probabilité de trouver un polynôme irréductible est grande.

Notons que si Q n'est pas irréductible, on a encore l'implication [si $f = 0$, alors $\forall (a_1, \ldots, a_n) \in (\mathbf{F}_p[X]/(Q))^n$, $f(a_1, \ldots, a_n) = 0$]. Ainsi, même lorsque le polynôme Q obtenu n'est pas irréductible, en effectuant les calculs dans $\mathbf{F}_p[X]/(Q)$ on ne risque pas de décréter à tort que f est non nul. Les cas où Q est réductible ne gênent donc pas. Il suffit alors de répéter toute la procédure $O(q)$ fois pour avoir une grande probabilité de tomber sur un polynôme irréductible et pouvoir conclure.[1]

2.2.3 Conclusion

Pour résumer, voici les opérations qu'effectue notre algorithme Co-RP, sur l'entrée (C, d). On appelle f le polynôme calculé par C.

- Construire le circuit calculant séparément chaque composante homogène de f. On appelle f_i la composante de degré i.
- Tester si $f_i = 0$ pour tout $i > d$. Sinon rejeter.

[1] Si toutefois l'on souhaite vérifier que Q est irréductible sur $\mathbf{F}_p$, il existe un algorithme polynomial pour cela, par exemple l'algorithme de Berlekamp [2]. On pourra se référer à [7, 4.2.1.1] pour une exposition de cet algorithme. Notons également qu'il existe des algorithmes déterministes, polynomiaux en q, pour trouver un polynôme irréductible sur $\mathbf{F}_p$, par exemple dans [18].

– Accepter.

Les étapes pour tester si un polynôme $f(x_1, \ldots, x_n)$ est nul sur un corps de caractéristique nulle sont les suivantes. Notons que dans notre cas, nous pouvons majorer $\deg(f)$ par $\deg_{\mathrm{f}}(C)$.

- Répéter k fois :
  - choisir aléatoirement n entiers $(a_1, \ldots, a_n)$ de valeur absolue $\leq 2\deg(f)$;
  - calculer $f(a_1, \ldots, a_n)$.
- Si au cours de la boucle, un des points n'annulait pas f, alors f n'est pas nul ; sinon, décréter que f est nul.

La probabilité de se tromper est $\leq 2^{-k}$.

Enfin, si l'on se trouve sur un corps de caractéristique $p > 0$, les étapes pour tester si un polynôme $f(x_1, \ldots, x_n)$ est nul sont les suivantes.

- Choisir un nombre premier q tel que $p^q \geq 2\deg(f)$.
- Répéter q fois :
  - choisir aléatoirement un polynôme unitaire $Q \in \mathbf{F}_p[X]$ de degré q ;
  - on se place maintenant dans l'anneau $\mathbf{F}_p[X]/(Q)$;
  - répéter k fois :
    - choisir aléatoirement n points $(a_1, \ldots, a_n)$,
    - calculer $f(a_1, \ldots, a_n)$;
  - si au cours de la boucle, un des points n'annulait pas f, alors sortir car f n'est pas nul.
- Si f était nul sur tous les points testés, décréter que f est nul.

La probabilité de se tromper est $O(2^{-k})$.

2.2.3.1 Remarque On pourrait également s'intéresser au problème suivant :

$$\mathrm{DEG}' = \{(C, 1^d) : \deg(C) \leq d\},$$

c'est-à-dire que d est écrit en unaire, et on n'a plus de restriction sur C. Dans ce cas, on calcule chaque composante homogène de degré $\leq d$ séparément, et encore séparément toute «la composante» de degré $> d$. Il suffit alors de tester la nullité de la dernière «composante» de degré $> d$. Nous obtenons également un algorithme Co-RP pour ce problème. Notons que dans un corps de caractéristique nulle, il faut faire de l'arithmétique modulaire, car nous n'avons plus de restriction sur le degré du polynôme calculé par C.

2.2.4 Borne inférieure

Bien qu'on ait un algorithme Co-RP pour $\mathrm{DEG}_{\mathrm{b}}$ (on peut donc considérer ce problème facile), on aimerait obtenir une borne inférieure sur la complexité de ce problème. Nous montrons une borne inférieure linéaire évidente : tout algorithme doit lire le circuit jusqu'au bout pour conclure.

En effet, soit un circuit C calculant un polynôme de degré strictement positif. On appelle C_+ le circuit C auquel on ajoute 0 au résultat, et $C_\times$ le circuit C dont le résultat est multiplié par 0. Alors $\deg(C_+) > 0$ mais $\deg(C_\times) \leq 0$. Cependant, puisque la porte ajoutée est la dernière, tout algorithme correct doit lire le code du circuit jusqu'au bout pour différencier les entrées $(C_+, 0)$ de $(C_\times, 0)$, et prend ainsi un temps (au moins) linéaire.

2.2.4.1 Remarque Dans la version non bornée, on montrera plus loin que DEG est P-dur pour la réduction en espace logarithmique. Ce résulat repose sur une simulation des circuits booléens par des polynômes, ce qui requiert de ne pas borner de degré formel complet des circuits algébriques.

2.3 Calcul du coefficient d'un monôme

Nous étudions maintenant la complexité du problème CMN et de sa version bornée $\mathrm{CMN}_{\mathrm{b}}$. Nous montrons que ces deux problèmes sont hors de la hiérarchie polynomiale PH en caractéristique nulle (sauf si celle-ci s'effondre), que $\mathrm{CMN}_{\mathrm{b}}$ est dans $\mathrm{P}^{\#\mathrm{P}}$, et que CMN est dans PSPACE. En caractéristique $k > 0$, nous montrons que ces deux problèmes sont $\mathrm{Mod}_k\mathrm{P}$-complets.

2.3.1 Caractéristique nulle

Nous travaillons ici sur $\mathbf{Z}$, c'est-à-dire que nous ne faisons pas d'arithmétique modulaire. Notre étude repose en grande partie sur le permanent de matrices à coefficients 0 ou 1, à qui on a dédié la partie 2.3.1. En effet, on montre que $\mathrm{CMN_b}$ en caractéristique nulle est au moins aussi difficile que savoir si le permanent d'une matrice vaut une certaine valeur v, donc l'étude du permanent nous donnera une borne inférieure sur la complexité de $\mathrm{CMN_b}$ (et *a fortiori* de CMN).

Le langage Perm

Soit $\mathrm{Perm} = \{(M, v) : \mathrm{per}(M) = v\}$, où M est une matrice à coefficients dans $\{0, 1\}$, et v un entier naturel (ce langage apparaît dans [9]). Rappelons que la fonction qui, à une matrice à coefficients dans $\{0, 1\}$, associe son permanent, est #P-complète.

Soit L un langage de $\mathrm{P}^{\#\mathrm{P}[1]}$ (le [1] signifie que la machine de Turing n'effectue qu'un seul appel à l'oracle). Soit $\mathcal{T}$ la machine de Turing déterministe fonctionnant en temps polynomial correspondante, qui fait un appel à l'oracle pour calculer le permanent d'une matrice M à coefficients dans $\{0, 1\}$. Puisque $\mathcal{T}$ fonctionne en temps polynomial, la matrice M dont $\mathcal{T}$ demande le permanent a une taille $p(n) \times p(n)$, où p est un polynôme en la taille n de l'entrée. Ainsi, le permanent de M est majoré par $(p(n))!$.

Construisons une nouvelle machine de Turing non déterministe $\mathcal{N}$, avec oracle Perm, qui reconnaît le langage L. $\mathcal{N}$ simule $\mathcal{T}$ tant que $\mathcal{T}$ ne fait pas appel à l'oracle. Puis $\mathcal{N}$ devine un entier v entre 0 et $2^{p(n)^2} - 1 \geq (p(n))!$, ce qui est possible en temps polynomial puisque $p(n)^2$ est polynomial en n. Après ce choix non déterministe, $\mathcal{N}$ appelle l'oracle Perm sur l'entrée (M, v). Si la réponse de l'oracle est négative (c'est-à-dire $\mathrm{per}(M) \neq v$), alors on rejette. Sinon, on a trouvé le permanent de M ($\mathrm{per}(M) = v$) : on simule alors la machine $\mathcal{T}$ comme si son oracle avait répondu v comme valeur du permanent de M.

Ainsi, si $\mathcal{T}$ accepte, notre nouvelle machine $\mathcal{N}$ a exactement un chemin acceptant (celui correspondant à la bonne valeur v pour le permanent) ; sinon notre machine n'a pas de chemin acceptant. Puisque notre machine fonctionne en temps polynomial, on en déduit que notre langage L est dans $\mathrm{NP}^{\mathrm{Perm}}$, et même $\mathrm{UP}^{\mathrm{Perm}[1]}$: Perm est $\mathrm{P}^{\#\mathrm{P}[1]}$-dur pour les réductions Turing non-déterministes.

Rappelons maintenant le théorème de Toda ([19]) : $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. Ainsi, par ce qui précède, nous avons montré que

$$\mathrm{PH} \subseteq \mathrm{NP}^{\mathrm{Perm}}.$$

Cela implique que Perm n'est pas dans la hiérarchie polynomiale PH, sauf si cette dernière s'effondre. En effet, supposons que Perm soit contenu dans Σ_i^p. Alors $\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p}$ contient $\mathrm{NP}^{\mathrm{Perm}}$, donc contient PH : la hiérarchie s'effondre.

Note : en particulier, puisque Perm est dans $\mathrm{P}^{\#\mathrm{P}[1]}$, nous avons le corollaire suivant : si $\mathrm{P}^{\#\mathrm{P}[1]}$ $=$ PH, alors la hiérarchie polynomiale s'effondre.

Le langage CMN

Réduction de Perm Soit une matrice $M = (m_{i,j})$, pour $1 \leq i, j \leq n$, et soit un entier v. Considérons le polynôme en les n variables X_j $(1 \leq j \leq n)$ suivant (qui est classique) :

$$P_{(M,v)}(X_1, \ldots, X_n) = \prod_{j=1}^{n} (\sum_{i=1}^{n} m_{i,j} X_i) - v X_1 \cdots X_n.$$

Le coefficient du monôme $X_1 \cdots X_n$ dans ce polynôme est $\mathrm{per}(M) - v$. En effet, pour chaque permutation σ de $\{1, \ldots, n\}$, il sort du produit un terme de la forme $\prod_i m_{i,\sigma(i)}$, donc en tout, on obtient

$$(\sum_{\sigma} \prod_{i} m_{i,\sigma(i)}) - v.$$

À partir de l'entrée (M, v), on construit facilement en temps polynomial un circuit calculant $P_{(M,v)}$. Puis tester si le coefficient du monôme $X_1 \cdots X_n$ est nul fourni la réponse au test $\mathrm{per}(M) = v$. De plus, le degré de ce polynôme est n. Ainsi, Perm est plus facile que $\mathrm{CMN_b}$ pour la réduction polynomiale many-one.

On en déduit que $\mathrm{CMN_b}$ n'est pas dans la hiérarchie polynomiale PH, sauf si celle-ci s'effondre. En effet, si $\mathrm{CMN_b}$ était dans PH, alors Perm serait aussi dans PH (stabilité par réduction many-one), ce qui entraîne, on l'a vu, un effondrement de PH.

Borne supérieure

Version bornée Nous allons maintenant montrer que $\mathrm{CMN_b}$ est dans $\mathrm{P}^{\#\mathrm{P}}$. Pour résoudre ce problème, nous suivons encore une fois la thèse de Guillaume Malod. On utilise la version «circuits multiplicativement disjoints» de $\mathrm{CMN_b}$.

[15] montre que le coefficient d'un monôme dans le polynôme calculé par le circuit multiplicativement disjoint n'est autre que le nombre de «développements» (sous-circuit représentant un seul monôme, avec coefficient 1) du circuit de départ, et dont les variables sont celles du monôme.

Or vérifier si un sous-circuit est un développement correspondant au monôme concerné, est un problème polynomial. Ainsi, la fonction qui compte le nombre de développements correspondant au monôme concerné est dans #P. Dans $\mathrm{P}^{\#\mathrm{P}}$, on teste donc si le coefficient est nul ou non. Notons que l'on effectue un seul appel à l'oracle.

Version générale Le problème de cette construction est flagrant : elle ne fonctionne que pour des circuits dont le degré formel complet est borné par un polynôme en la taille du circuit.

On peut toutefois résoudre CMN en toute généralité, en espace polynomial. En effet, on utilise pour cela la caractérisation de PSPACE par des circuits booléens de profondeur polynomiale (mais éventuellement de taille exponentielle).

Grace à un circuit booléen, nous évaluons le polynôme calculé par chaque porte de notre circuit algébrique. Pour cela, nous codons chaque terme par son coefficient, chiffre par chiffre (cela prend au maximum un nombre simplement exponentiel de portes, puique chaque coefficient est $\leq 2^{2^n}$, où n est la taille du circuit), et par la suite de ses puissances (ce qui prend un nombre polynomial de portes supplémentaires). Multiplier deux termes se fait alors en profondeur polynomiale, ainsi qu'additionner deux termes (profondeur $\log(2^{n^{O(1)}})$).

En profondeur polynomiale, on peut donc calculer chaque chiffre du coefficient du monôme qui nous intéresse. Il suffit alors de tester s'il est nul, en faisant (diviser pour régner) un OU de tous les chiffres. Ainsi, nous avons un algorithme PSPACE permettant de décider si le coefficient du monôme dans le circuit algébrique est nul.

Conclusion

Pour résumer, on a montré que CMN est dans PSPACE, mais la version bornée $\mathrm{CMN_b}$ est dans $\mathrm{P}^{\#\mathrm{P}[1]}$. On a également montré que ni CMN ni $\mathrm{CMN_b}$ ne sont dans la hiérarchie polynomiale, sauf si celle-ci s'effondre.

2.3.2 Caractéristique non nulle

Cette fois, on se place sur un corps de caractéristique k non nulle, c'est-à-dire que nous faisons de l'arithmétique modulo k où k est un nombre premier. Plus précisément, nous nous intéressons au problème

$$\mathrm{CMN}_k = \{(C, m) : \mathrm{coef}_C(m) \equiv 0 \bmod k\}.$$

Dans ce cas, nous verrons que l'on peut tester la nullité du coefficient d'un monôme dans $\mathrm{Mod}_k\mathrm{P}$. Notons que le permanent se calcule facilement modulo 2 (car alors il coïncide avec le déterminant), donc on ne peut plus utiliser le permanent pour montrer des résultats de complétude, si l'on ne veut pas négliger la caractéristique 2. On prendra plutôt le problème de compter le nombre de circuits hamiltoniens d'un graphe, qui est complet quelque soit la caractéristique.

Algorithme

Nous nous servons de la suite de polynômes (G_n) complète pour $\mathrm{V}\Sigma^0\mathrm{P}$ de [15, chap. 5] (voir la partie 1.4 de ce rapport). Dans sa thèse, Malod a calculé les coefficients des monômes de G_n. Pour cela, il construit un graphe orienté associé au monôme à étudier, et exprime le coefficient du monôme en fonction du nombre d'arêtes gauches et droites allant du sommet i au sommet j ($A_{i,j}$ et $B_{i,j}$ respectivement). On note k_i le nombre d'arêtes arrivant au sommet i. Le coefficient du monôme est alors le produits pour i de 1 à n de $a_i b_i$, où a_i et b_i sont de la forme (on utilise la notation anglo-saxone des coefficients binomiaux pour la lisibilité)

$$a_i = \binom{k_i}{A_{i,i-1}} \binom{k_i - A_{i,i-1}}{A_{i,i-2}} \cdots \binom{k_i - A_{i,i-1} - \ldots - A_{i,-n+1}}{A_{i,-n}}$$

$$b_i = \binom{k_i}{B_{i,i-1}} \binom{k_i - B_{i,i-1}}{B_{i,i-2}} \cdots \binom{k_i - B_{i,i-1} - \ldots - B_{i,-n+1}}{B_{i,-n}}$$

Grâce au résultat d'Édouard Lucas ([14, pp. 229–230]), on peut calculer en temps polynomial les gros coefficients binomiaux modulo k, en décomposant leur composantes en base k :

2.3.2.1 Proposition (Lucas,1878) Soit k un nombre premier. Soient $N = \sum_{i=0}^{n} n_i k^i$ et $M = \sum_{i=0}^{n} m_i k^i$ deux entiers décomposés en base k. Alors

$$\binom{N}{M} \equiv \prod_{i=0}^{n} \binom{n_i}{m_i} \mod k.$$

Ainsi, le coefficient de tout monôme de G_n se calcule en temps polynomial modulo k. Il ne reste plus qu'à appliquer ce résultat à tout polynôme de $\mathrm{V}\Sigma^0\mathrm{P}$. Puisque (G_n) est $\mathrm{V}\Sigma^0\mathrm{P}$-complète, le circuit C est obtenu comme la projection de $G_{t(|C|)}$: $C = \sigma(G_{t(|C|)})$. Cette projection σ (qui associe aux variables de $G_{t(|C|)}$ des variables de C ou des constantes de $\{-1, 0, 1, 2\}$) se calcule en temps polynomial car (G_n) simule une suite de circuits génériques. Si maintenant m est le monôme dont on veut connnaître la nullité du coefficient, on a :

$$\mathrm{coef}_C(m) = \sum_{m', \sigma(m')=m} \mathrm{coef}_{G_{t(n)}}(m').$$

On a vu qu'en temps polynomial, on pouvait calculer $\mathrm{coef}_{G_{t(n)}}(m')$, et aussi déterminer si $\sigma(m') = m$. Ainsi, cette somme est $\mathrm{Mod}_k\mathrm{P}$, ce qui montre que $\mathrm{CMN}_k \in \mathrm{Mod}_k\mathrm{P}$.

Borne inférieure

Nous montrons maintenant que CMN_k est dur pour $\mathrm{Mod}_k\mathrm{P}$. Puisque «Circuit Hamiltonien» (la fonction qui compte le nombre de circuits hamiltoniens d'un graphe) est #P-complète pour les réductions parsimonieuses ([23]), elle est aussi complète modulo k, c'est-à-dire que «Circuit Hamiltonien modulo k» est $\mathrm{Mod}_k\mathrm{P}$-complet (puisque #SAT se réduit de manière parsimonieuse à «Circuit Hamiltonien», on en déduit que $\mathrm{Mod}_k\mathrm{SAT}$ se réduit à «Circuit Hamiltonien modulo k»). Or, de même que pour le permanent, le nombre de circuits hamiltoniens d'un graphe s'obtient comme coefficient d'une certaine suite de polynômes $(K_n) \in \mathrm{V}\Sigma^0\mathrm{P}$ à partir de sa matrice d'adjacence $(x_{i,j})$ (*cf* [15, lemme 14]) :

$$HC_n(x_{1,1} \ldots, x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)} = \mathrm{coef}_m(K_n),$$

où la somme est prise sur tous les cycles σ de longueur n de $\{1, \ldots, n\}$, et où K_n est de degré formel complet polynomial.

La définition de K_n est constructive, et on peut créer en temps polynomial un circuit calculant K_n. Ainsi, savoir si le nombre de circuits hamiltoniens d'un graphe est nul modulo k revient à savoir si le coefficent d'un monôme d'un polynôme calculé par un circuit constructible en temps polynomial est nul modulo k. Donc CMN_k est $\mathrm{Mod}_k\mathrm{P}$-dur.

2.3.2.2 Remarque Le théorème de Toda montre que $\mathrm{PH} \subseteq \mathrm{BPP}^{\oplus\mathrm{P}}$. [20] le généralise à

$$\mathrm{PH} \subseteq \mathrm{BPP}^{\mathrm{Mod}_k\mathrm{P}}$$

pour tout k, ce qui implique que CMN_k n'est pas dans la hiérarchie polynomiale sauf si elle s'effondre.

2.4 Le cas général

2.4.1 DEG est P-dur

Nous montrons que DEG est plus difficile que le Problème de la Valeur d'un Circuit (PVC). PVC est le langage consistant des couples (B, b) où B est un circuit *booléen*, sans variable en entrée, et b est la valeur de la sortie. PVC est un problème P-complet pour les réductions en espace logarithmique. Dans un circuit algébrique C, on va coder la sortie du circuit B.

Soit C_0 le circuit algébrique (sans variable) qui simule B par des polynômes. Le circuit C est obtenu comme suit : on ajoute une entrée x à C_0 (x est une variable), puis on multiplie x par la sortie de C_0 (qui vaut $b \in \{0, 1\}$). C calcule le polynôme bx, dont le degré est 1 si $b = 1$, et ≤ 0 sinon. Ainsi, DEG sur l'entrée $(C, 0)$ résoud PVC : DEG est P-dur pour la réduction en espace logarithmique.

2.4.2 Relation entre CMN et DEG

Bien entendu, les deux problèmes que nous avons étudiés jusqu'à présent sont liés, puisqu'ils concernent tous deux la nullité de certains coefficients : celui d'un monôme précis pour CMN, et ceux de chaque monôme d'une composante homogène pour DEG. Nous avons le lien suivant entre la complexité des deux problèmes :

$$\mathrm{DEG} \in \text{Co-NP}^{\mathrm{CMN}},$$

où Co-NP est l'ensemble des langages dont le complémentaire est dans NP.

En effet, $\deg(C) \leq d$ est équivalent à (tout monôme de degré $> d$ a un coefficient nul), ce qui se teste dans Co-NP avec appels à l'oracle CMN.

Ainsi, en caractéristique $k > 0$ (et sans restriction sur le degré formel complet du circuit), on en déduit que

$$\mathrm{DEG} \in \text{Co-NP}^{\mathrm{Mod}_k\mathrm{P}}.$$

2.5 Récapitulatif

Afin de résumer les résultats de ce chapitre, nous les avons regroupés dans le tableau suivant (où on suppose que la hiérarchie polynomiale est stricte).

	Caractéristique nulle		Caractéristique $k > 0$	
	Borne inférieure	Borne supérieure	Borne inférieure	Borne supérieure
$\mathrm{CMN_b}$	Hors de PH	$\mathrm{P}^{\#\mathrm{P}[1]}$	$\mathrm{Mod}_k\mathrm{P}$	
CMN	Hors de PH	PSPACE		
$\mathrm{DEG_b}$	Linéaire	Co-RP	Linéaire	Co-RP
DEG	P	PSPACE	P	$\text{Co-NP}^{\mathrm{Mod}_k\mathrm{P}}$

Conclusion

Dans ce rapport, nous avons abordé deux aspects du calcul de polynômes $f \in K[x_1, \ldots, x_n]$ par des circuits algébriques.

- Le premier chapitre a étudié une hiérarchie de classes de complexité du point de vue de la complexité algébrique, dans l'esprit de la hiérarchie polynomiale. Nous calculons des suites de polynômes définis par de grosses sommes ou de gros produits : une séparation de ces classes montrerait que ces sommes ou ces produits augmentent véritablement la puissance de calcul, et un effondrement aurait de nombreuses conséquences inattendues. Enfin, nous nous servons des gros produits pour créer un circuit de séparation d'hyperplans, ce qui nous donne un résultat de transfert sur les classes algébriques P et NP.
- Dans le second chapitre, nous avons étudié la manipulation par machine de Turing booléenne de polynômes donnés par des circuits. Nous envisageons deux problèmes, le calcul du degré d'un polynôme et le calcul du coefficient d'un monôme. Nous avons montré que dans un cas restreint, le degré se calcule efficacement par un algorithme probabiliste, mais que le coefficient du monôme est difficile. Des résultats pertinents dans le cas général (notamment pour le calcul du degré) semblent plus durs à atteindre.

Il serait intéressant d'étudier plus en détail la hiérarchie des $(\mathrm{V}\Sigma^i\mathrm{P})$ et $(\mathrm{V}\Pi^i\mathrm{P})$: séparer des classes ou montrer que des classes coïncident auraient des conséquences très importantes.

Par ailleurs, nous utilisons les gros produits pour séparer des hyperplans. Il semble naturel de vouloir séparer des hypersurfaces, puisque nos circuits sont capables de calculer des polynômes. Un résultat sur la séparation d'hypersurfaces permettrait peut-être d'énoncer des résultats de transfert sur $\mathrm{NP}_{(K,+,-,\times,=)}$ et $\mathbb{P}_{(K,+,-,\times,=)}$, c'est-à-dire d'utiliser la multiplication même dans NP, ce qui n'est pas le cas du résultat du chapitre 1.

Concernant la complexité des problèmes CMN et DEG, du travail reste à accomplir afin d'améliorer le tableau récapitulatif du chapitre 2. Par exemple, on pourrait essayer d'améliorer la borne supérieure PSPACE de CMN en caractéristique nulle. Ou encore de limiter l'écart entre la borne inférieure (P) et la borne supérieure (PSPACE ou $\mathrm{Co\text{-}NP}^{\mathrm{Mod}_k\mathrm{P}}$ selon la caractéristique) de DEG en version non bornée.

Notons enfin que sans la thèse de Guillaume Malod ([15]) et la perspicacité de Pascal Koiran, beaucoup de résultats n'auraient certainement pas eu leur place dans ce rapport. Je les remercie vivement, ainsi que les autres membres de l'équipe.

Bibliographie

[1] Richard Beigel and John Gill. Counting classes : Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1) :3–23, 1992.

[2] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Tech. Journal*, 46 :1853–1859, 1967.

[3] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. Algebraic settings for the problem "P $\neq$ NP ?". *The Mathematics of Numerical Analysis*, 42 :125–144, 1996.

[4] Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation*. Springer, 1998.

[5] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines. *Bulletin of the Amer. Math. Soc.*, 21 :1–46, 1989.

[6] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000.

[7] James H. Davenport, Yvon Siret, and Évelyne Tournier. *Computer Algebra*. Academic Press, second edition, 1993.

[8] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. EATCS. Springer, 2002.

[9] Valentine Kabanets and Russel Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Electronic Colloquium on Computational Complexity*, 55, 2002.

[10] Erich Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM*, 35(1) :231–264, January 1988.

[11] Neal Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag, 1987.

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

[13] Pascal Koiran. Valiant's model and the cost of computing integers. Technical report, LIP, January 2004.

[14] Édouard Lucas. Théorie des fonctions numériques simplement périodiques. *American Journal of Mathematics pure and applied*, 1 :184–240 and 289–321, 1878.

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

[16] Chistos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

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

[18] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54 :435–447, 1990.

[19] Seinosuke Toda. On the computational power of PP and $\oplus$P. In *Proc. 30th IEEE Symposium on the Foundations of Computer Science*, pages 514–519, 1989.

[20] Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21 :316–328, 1992.

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

[22] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2) :189–201, 1979.

[23] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8 :410–421, 1979.

Annexe A

Autour de l'hypothèse FPSPACE $\subseteq$ GapP/poly

Dans le premier chapitre, nous avons utilisé l'hypothèse FPSPACE $\subseteq$ GapP/poly. Pour essayer de donner une intuition un peu meilleure de cette hypothèse à partir de classes de langages plutôt que de classes de fonctions, nous étudions dans cette partie ce qu'elle implique, et ce qui l'implique. Si A est un langage sur l'alphabet Σ, et $f : \mathbf{N} \to \mathbf{N}$ une fonction, pour $x \in \Sigma^*$ on définit $\#A_f(x)$ par le cardinal de l'ensemble $\{y \in \Sigma^{f(|x|)} : (x, y) \in A\}$. Nous prendrons dans ce qui suit $\Sigma = \{0, 1\}$.

A.1 Une conséquence de FPSPACE $\subseteq$ GapP/poly

Nous allons montrer le résultat suivant :

$$\text{FPSPACE} \subseteq \text{GapP/poly} \Rightarrow \text{PSPACE} \subseteq \oplus\text{P/poly}.$$

Soit L un langage de PSPACE. On définit une fonction $f : \{0,1\}^* \to \{0,1\}$ par $f(x) = 1$ si et seulement si $x \in L$. Cette fonction est dans FPSPACE, puisqu'il suffit de tester l'appartenance à L. Ainsi, par hypothèse, f est dans GapP/poly. Par définition, il existe donc une fonction de conseil polynomiale u, deux langages A et B de P, et deux polynômes p et q, tels que

$$f(x) = \#A_p(x, u(|x|)) - \#B_q(x, u(|x|)).$$

Puisque notre fonction f prend ses valeurs dans $\{0, 1\}$, on en déduit que $\#A_p(x, u(|x|)) + \#B_q(x, u(|x|))$ est impair si et seulement si $f(x) = 1$, si et seulement si $x \in L$. Or le langage C définit par l'ensemble des mots (y, a, u, v), où a est de taille 1, u de taille $p(|y|)$, et v de taille $q(|y|)$, tels que

- si $a = 0$ alors $v = 0 \ldots 0$ et $(y, u) \in A$;
- si $a = 1$ alors $u = 0 \ldots 0$ et $(y, v) \in B$,

vérifie $\#C_{p+q+1}(y) = \#A_p(y) + \#B_q(y)$, et est dans P. Ainsi,

$$L = \{x : \#C_{p+q+1}(x, u(|x|)) \text{ impair}\},$$

donc $L \in \oplus$P/poly.

Bien entendu, la version uniforme FPSPACE $\subseteq$ GapP $\Rightarrow$ PSPACE $\subseteq \oplus$P est vraie, et se montre de la même manière.

A.1.1 Remarque L'inclusion suivante est vraie (*cf* [8, cor. 4.10]) :

$$\text{PH} \subseteq \oplus\text{P/poly}.$$

A.2 Une conséquence de PSPACE $\subseteq$ UP/poly

La réciproque du résultat précédent n'est pas claire, mais nous allons montrer le résultat suivant :

$$\text{PSPACE} \subseteq \text{UP/poly} \Rightarrow \text{FPSPACE} \subseteq \text{GapP/poly}.$$

En fait, pour simplifier la preuve, nous allons montrer la version uniforme du résultat, c'est-à-dire

$$\text{PSPACE} \subseteq \text{UP} \Rightarrow \text{FPSPACE} \subseteq \text{GapP}.$$

La preuve du cas non-uniforme est essentiellement la même.

Soit f une fonction de FPSPACE. Supposons f positive. Alors il existe un polynôme p tel que

$$f(x) = \sum_{i=0}^{p(|x|)} f_i(x) 2^i,$$

où les fonctions f_i, à valeurs dans $\{0,1\}$, sont telles que $A := \{(x,i) : f_i(x) = 1\}$ est dans PSPACE (pour simplifier, on supposera sans perte de généralité que i est codé en unaire : ainsi, la taille de i est $p(|x|)$). Par hypothèse, on a donc $A \in$ UP. Ainsi, il existe un langage B de P, et un polynôme q, tels que

- si $(x,i) \in A$ alors il existe un unique y, de taille $q(|x|)$, tel que $(x,i,y) \in B$;
- si $(x,i) \notin A$ alors il n'existe aucun y de taille $q(|x|)$ tel que $(x,i,y) \in B$.

On définit le langage C de P comme l'ensemble des mots (x,i,y,v), où y est de taille $q(|x|)$, v est de taille $p(|x|)$ et code en binaire un entier $\leq 2^i - 1$ (*i.e.* toutes ses $p(|x|) - i + 1$ premières lettres sont 0), et tel que $(x,i,y) \in B$. Puisque A est dans UP, le cardinal de $\{(y,v) : (x,i,y,v) \in C\}$ est $f_i(x)2^i$, pour tout i. Ainsi, $\#C_{2p+q}(x) = \sum_{i=0}^{p(|x|)} f_i(x)2^i = f(x)$. Ainsi, f est dans $\#$P.

Si f prend aussi des valeurs négatives, en décomposant f selon ses parties positive et négative, on voit que f est alors la différence de deux fonctions de $\#$P, c'est-à-dire une fonction de GapP.

A.3 Résumé

Pour résumer, nous avons montré les implications suivantes :

$$\text{PSPACE} \subseteq \text{UP/poly} \Rightarrow \text{FPSPACE} \subseteq \text{GapP/poly} \Rightarrow \text{PSPACE} \subseteq \oplus\text{P/poly}.$$