N° d'ordre : 430
N° attribué par la bibliothèque : 07ENSL0 430

Thèse

en vue d'obtenir le grade de

docteur de l'Université de Lyon – École Normale Supérieure de Lyon

spécialité : informatique

Laboratoire de l'Informatique du Parallélisme

École doctorale de Mathématiques et d'Informatique Fondamentale

présentée et soutenue publiquement le 6 décembre 2007 par

Sylvain Perifel

Problèmes de décision et d'évaluation en complexité algébrique

Directeur de thèse : Pascal Koiran

Après avis de :	Eric Allender	Membre / rapporteur
	Peter Bürgisser	Membre / rapporteur
	Bruno Poizat	Rapporteur

Devant la commission d'examen formée de :

Eric Allender	Membre / rapporteur
Peter Bürgisser	Membre / rapporteur
Arnaud Durand	Membre
Pascal Koiran	Membre / directeur de thèse
Michel de Rougemont	Membre

Remerciements

« Non, plus j'y pense, plus ces remerciements au jury, à l'équipe, au public me paraissent… une façon de parler… un détour obligatoire pour ne pas avoir à se remercier officiellement soi-même. Il est absolument impossible de se remercier soi-même quand on n'est pas ministre. »

Daniel Pennac, *Merci*.

Je me demandais comment j'allais commencer les remerciements de ma thèse. Et bien je me demande toujours mais maintenant c'est fait. Ma pensée va tout d'abord à ma famille et en particulier à Lucie qui me supporte, sans trop râler, depuis déjà un bon paquet d'années.

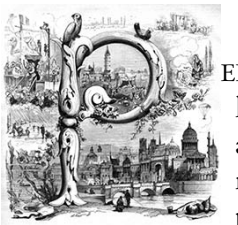
Je remercie Pascal, un directeur de thèse aussi lumineux que disponible. Cette thèse serait bien plus vide sans ses idées incessantes. Les rapporteurs auraient donc pu en lire moins mais tant pis. Qu'ils soient eux aussi remerciés, Bruno, Eric et Peter, pour leur lecture assidue et leurs commentaires, ainsi que pour le voyage qui les a mené jusqu'à la soutenance. Arnaud et Michel ont également dû faire le déplacement, merci !

Merci à Natacha pour sa bonne humeur et ses histoires de machine à pain — entre autre —, à Jacques (son Ardèche et sa pipe) et Marianne pour les discussions, les conseils et les manifestations. À Emmanuel, Emmanuelle, Victor, Sylvain S. et C., Stéphane, Laurent, Florent, Damien, Jean-Baptiste, Vincent pour la bonne ambiance dans les bureaux. À Guillaume M. pour des discussions plus scientifiques, Guillaume T. et Laurent B. pour des discussions plus politiques. À Sophie pour son amabilité. À ceux qui ont fait l'effort de venir à la soutenance, voire au pot.

Et à tous ceux que j'oublie.

Table des matières

Introduction



PERMETTONS-NOUS de débuter cette introduction par une digression sur la complexité booléenne. La formalisation de la notion de calcul dans les années 1930, par Gödel, Church, Turing, Kleene et Post notamment, a naturellement posé la question de ce qui était calculable. Ce n'est que plus tard que l'étude de l'efficacité des algorithmes prit le devant de la scène et donna naissance à la théorie de la complexité. L'idée générale de mesurer le temps et l'espace de calcul en fonction de la taille de l'entrée est due à Hartmanis et Stearns dans les années 1960. Le contexte le plus familier est alors celui des problèmes de décision, c'est-à-dire des langages sur l'alphabet $\{0,1\}$ (ou, plus généralement, sur un alphabet fini). La question traditionnelle est d'évaluer la complexité du meilleur algorithme pour décider ces langages, la notion d'algorithme étant formalisée par exemple par les machines de Turing : une tête de lecture et d'écriture, n'ayant elle-même qu'une mémoire finie, lit, écrit et se déplace sur un ou plusieurs rubans infinis dont les cellules soit sont vides soit contiennent un symbole 0 ou 1.

On parvint relativement vite à définir des classes de complexité pour ranger les problèmes, donnant du sens à la classification empirique de ce que l'on peut résoudre efficacement ou non : c'est cette branche « structurelle » qui nous intéresse ici. En complexité, à tort ou à raison, on considère en général que les problèmes que l'on sait résoudre efficacement sont ceux qui possèdent un algorithme s'exécutant en temps polynomial en fonction de la taille de l'entrée, c'est-à-dire ceux de la classe P (si toutefois l'on met de côté les algorithmes probabilistes). D'autres classes de problèmes ont également été définies et une question majeure est de savoir si elles sont différentes de P, c'est-à-dire si leurs problèmes possèdent des algorithmes efficaces. En particulier, beaucoup de problèmes naturels appartiennent à la classe NP, d'où la fameuse question ouverte « $\mathsf{P} = \mathsf{NP}\,?$ ». Une avancée majeure fut les résultats de Cook de NP-complétude du problème de satisfaisabilité des formules booléennes et, indépendamment, de Levin pour la NP-complétude d'un problème de pavage, suivis de la preuve de complétude d'un nombre impressionnant d'autres problèmes. En deux mots, certains problèmes de la classe NP (et la plupart de ceux sur lesquels on se casse les dents) sont ainsi au moins aussi difficiles que tout autre problème de NP. Ces résultats donnaient un premier cadre théorique à la frustration des algorithmiciens de ne pas trouver d'algorithmes efficaces pour ces problèmes.

Ces questions structurelles, dont la plus célèbre est de savoir si les classes P et NP

coïncident, ont pris une grande importance car la plupart d'entre elles restent ouvertes malgré tous les efforts et les années passés. On a donc vu une floraison de nouvelles classes et techniques dans l'espoir de contourner ces obstacles. C'est une des raisons pour lesquelles, outre son intérêt intrinsèque, la théorie de la complexité booléenne s'est énormément étoffée depuis son apparition. Le nombre de classes de complexité a explosé et le peu de résultats de séparation ou d'égalité suggère que le nombre de questions ouvertes en a fait de même…*

Complexité algébrique

C'est donc dans ce contexte qu'est apparue la complexité algébrique. La motivation principale était d'abstraire le modèle booléen afin de travailler sur des ensembles plus riches que $\{0,1\}$ car, par exemple, les machines de Turing booléennes semblaient peu adaptées pour manipuler des nombres réels. En bonus, l'espoir était d'utiliser les outils mathématiques beaucoup plus développés sur ces structures pour enfin pouvoir répondre à des questions analogues à celles du cas booléen.

Puisqu'il s'agit d'un cadre plus simple et d'un niveau d'abstraction plus élevé pour de nombreux problèmes numériques, une première approche, la conception d'algorithmes efficaces, a été très active dans ce domaine. On pourra citer en exemple la manipulation de matrices (multiplication, déterminant, rang, etc.) ou de polynômes (évaluation, pgcd, test de nullité, etc.). D'autre part, une deuxième approche plus spécifique à la complexité s'est attachée à prouver des bornes inférieures sur ces problèmes, c'est-à-dire un nombre minimum d'opérations que mettrait l'algorithme le plus rapide pour résoudre le problème. L'utilisation de la structure mathématique sous-jacente a permis quelques progrès : à titre d'exemples simples, le degré des polynômes, la dimension des espaces vectoriels ou le degré de transcendance sont des invariants utiles dans les preuves de bornes inférieures. L'objectif ultime est alors de trouver une borne supérieure (c'est-à-dire un algorithme) et une borne inférieure qui coïncident : dans ce cas, nous avons la preuve que l'algorithme fourni est le meilleur possible. Cependant, rares sont les cas où nous avons une telle situation car, on s'en doute, montrer une bonne borne inférieure est très difficile puisqu'il s'agit d'une propriété concernant tous les algorithmes possibles. On pourra se référer au livre de Bürgisser, Clausen et Shokrollahi [16] pour aller au-delà de cette courte introduction.

Bien entendu, une troisième approche concerne les questions structurelles similaires à celles de la complexité booléenne et a également reçu beaucoup d'attention ; c'est elle qui nous intéressera plus particulièrement. Valiant proposa d'abord son modèle pour les problèmes d'évaluation, puis celui de Blum, Shub et Smale (BSS) apparut pour les problèmes de décision. C'est ainsi qu'ont été étudiés les analogues algébriques de P et NP et qu'ont également vu le jour des théories algébriques de la NP-complétude. Dans le cas des problèmes d'évaluation, il s'agit de calculer des polynômes ; en revanche, comme dans le cas booléen, un problème de décision est un langage, mais l'alphabet est quelconque, généralement infini, généralement un corps tel $\mathbb{R}$ ou $\mathbb{C}$.

Las, de la même manière que précédemment et bien que le nombre de classes de complexité introduites soit sans commune mesure, les questions principales restent ouvertes

*On pourra compléter ce bref panorama historique de la théorie de la complexité booléenne par la lecture de Fortnow et Homer [26].

et certains théorèmes de transfert montrent qu'elles sont aussi difficiles que dans le cas booléen. Dans cette thèse, on se propose de comparer ces questions non pas avec le cas booléen, mais entre les modèles algébriques de Valiant et de BSS que nous présentons brièvement ci-dessous. Puisque nous manipulerons des polynômes, les outils mathématiques à notre disposition sont potentiellement nombreux et viendront principalement de la géométrie algébrique. En particulier, nous utiliserons des algorithmes provenant de l'étude de l'élimination des quantificateurs.

Nous devrions mentionner que plusieurs autres modèles généralisant le calcul à des structures plus riches que $\{0,1\}$ ont vu le jour. On notera en effet que les deux modèles mentionnés précédemment sont adaptés seulement à la manipulation de polynômes (d'où l'adjectif « algébrique ») et on ne peut, par exemple, calculer la fonction exponentielle qui paraît pourtant très naturelle. Notamment, bien avant les modèles algébriques ci-dessus, des modèles de calcul analogique où l'on peut intégrer et différencier des fonctions ont été introduits. Dans cette thèse, nous nous contenterons d'étudier les modèles BSS et de Valiant, dont nous donnons maintenant une brève description.

Modèle de Valiant

Il s'agit vraisemblablement d'un des modèles les plus simples et les plus élégants. On se place sur un corps K quelconque et on calcule des familles de polynômes à coefficients dans K grâce à des familles de circuits arithmétiques, c'est-à-dire possédant des portes d'addition et de multiplication.

L'analogue de la classe P s'appelle VP et il s'agit essentiellement des familles de polynômes calculées par une famille de circuits arithmétiques de taille polynomiale. Une sorte d'analogue de la classe NP s'appelle VNP et il s'agit de sommes de taille exponentielle de familles VP.[†]

Comme dans le cas booléen, il y a une théorie de la VNP-complétude et l'une des questions principale est de savoir si VP et VNP coïncident.

On aura pu le constater, les classes de Valiant ci-dessus sont non-uniformes : on a besoin d'un circuit distinct pour chaque polynôme de la famille. Cependant, pour pouvoir comparer avec le modèle BSS qui, lui, est uniforme, nous avons choisi dans ce document de traiter surtout des variantes uniformes des classes de Valiant.

Modèle BSS

On se place également sur un corps K mais cette fois on reconnaît des langages sur l'alphabet K grâce à des familles de circuits algébriques, c'est-à-dire possédant des portes d'addition, de multiplication et de tests. On peut voir un circuit comme un algorithme travaillant sur des entrées de taille fixée et manipulant directement des éléments de K, pouvant les additionner, les multiplier et les comparer en une opération. En particulier, toutes les opérations sont exactes donc on ne se soucie pas de la précision des calculs. Par ailleurs, comme on l'a dit précédemment, le modèle BSS est uniforme donc une condition d'uniformité est imposée sur la famille de circuits. Il s'avère que la définition originale de Blum, Shub et Smale est en termes de machines travaillant sur $\mathbb{R}$, mais dans

[†]En réalité, comme nous le verrons, VNP est certainement plus proche de $\#\mathsf{P}$ que de NP.

cette thèse nous utiliserons de manière équivalente des familles (uniformes) de circuits algébriques afin de souligner les liens avec le modèle de Valiant.

On notera P_K l'analogue de la classe P sur K, qui est donc l'ensemble des langages sur K reconnus par une famille (uniforme) de circuits algébriques de taille polynomiale. La classe NP_K est alors la version existentielle de P_K, comme dans le cas booléen sauf que le témoin est un mot de K.

Comme dans le cas booléen, il y a une théorie de la NP_K-complétude et l'une des questions principales est de savoir si P_K et NP_K coïncident.

Dans cette thèse, nous argumentons qu'il est plus sage de commencer par étudier le modèle de Valiant, car le modèle est plus simple et les réponses ne sont pas plus difficiles. Les résultats que nous obtenons comparent en effet les grandes questions ouvertes dans ces modèles : si nous parvenions à séparer telles classes dans le modèle BSS alors la séparation serait vraie également dans le modèle de Valiant. Les classes principales que nous étudions sont les analogues algébriques de P (temps déterministe polynomial), de NP (temps non-déterministe polynomial) et de PSPACE (espace polynomial).

Contenu de la thèse

Dans un souci d'originalité, cette thèse débute par un chapitre définissant les notions principales utilisées dans le document. On y traite de circuits, qu'ils soient booléens, arithmétiques ou même algébriques, et de classes de complexité, concernant des langages ou des familles de polynômes. Les notions exposées sont nombreuses mais assez standard. Ce chapitre traite également de l'uniformité des circuits.

Afin de se familiariser avec les circuits, nous proposons dans le deuxième chapitre une étude algorithmique de la manipulation des circuits arithmétiques par des machines de Turing. En effet, ces circuits peuvent encoder de manière très concise des polynômes et il est souhaitable de savoir si l'on parvient alors à les utiliser efficacement. Ainsi la complexité de deux problèmes naturels est étudiée. Le but du premier est de décider si le coefficient d'un monôme du polynôme calculé par un circuit est nul. Le second concerne le calcul du degré du polynôme calculé par un circuit. Nous verrons que le premier semble difficile et que ces deux problèmes se placent dans la hiérarchie de comptage.

Bien qu'il utilise les résultats de Malod [57] sur les classes de Valiant, ce chapitre n'est pas exactement dans l'esprit des suivants. En effet, nous considérons ici les circuits en tant qu'objets eux-mêmes manipulés par des machines, alors que par la suite nous les étudierons vraiment en tant que modèle de calcul — notamment, nous parlerons de classes de complexité algébrique, à la BSS ou à la Valiant. À l'origine, seules les classes VP et VNP ont été étudiées dans le modèle de Valiant. En se rappelant que les familles VNP sont des sommes de taille exponentielle de familles VP, il semble dès lors assez naturel d'étudier également des produits de taille exponentielle.

Nous obtenons ainsi une classe, appelée $\mathsf{VΠP}$, de familles de polynômes qui sont des produits de taille exponentielle de familles VP. C'est l'objet d'étude du chapitre suivant. Plus précisément, on s'intéresse à la question de savoir si ces produits de taille exponentielle peuvent être calculés par des circuits de taille polynomiale. Tout d'abord, nous montrons que répondre négativement à cette question revient à séparer VP et VNP, ce qui semble hors de portée pour l'instant. Pour obtenir un tel résultat, on utilise l'interpolation de Lagrange et les résultats récents de Bürgisser concernant le calcul de sommes

et de produits dans la hiérarchie de comptage. Puis le résultat principal du chapitre relie le problème initial (est-ce que les produits exponentiels sont calculables par des circuits de taille polynomiale) à une question dans le modèle BSS : peut-on séparer P et NP sur un corps K grâce à un problème de NP sans multiplication (on suspecte en effet certains de ces problèmes d'être hors de P) ? Notre résultat prouve que pour répondre à cette question, il faudrait d'abord montrer que les produits de taille exponentielle ne sont pas calculables par des circuits arithmétiques de taille polynomiale. En particulier, cela impliquerait que VP et VNP diffèrent. Il s'agit de notre premier résultat de transfert : si nous pouvons séparer P et NP dans le modèle BSS grâce à un problème sans multiplication, alors nous aurions également séparé les analogues de P et NP dans le modèle de Valiant.

Indignés devant le sort de la multiplication qui avait alors disparu de NP dans le modèle BSS, les chapitres d'après tentent de réparer cette injustice en introduisant une nouvelle classe de familles de polynômes dans le modèle de Valiant, VPSPACE. Essentiellement, il s'agit de l'ensemble des familles de polynômes dont les coefficients sont calculables en espace polynomial. L'intérêt est de pouvoir manipuler des hypersurfaces grâce à des algorithmes connus fonctionnant en espace polynomial. Nous passons une bonne partie du chapitre 4 à définir cette classe et prouver les premiers résultats.

La fin du chapitre est consacrée à la question de savoir si les familles de VPSPACE possèdent des circuits de taille polynomiale. Deux approches sont suivies : d'abord, donner des hypothèses équivalentes mais utilisant des classes plus connues comme VP, VNP, P et PSPACE. Nous verrons alors qu'il est très improbable que VPSPACE ait des circuits de taille polynomiale. L'autre approche consiste à utiliser la complexité booléenne pour montrer que, si une hypothèse issue de la complexité de Kolmogorov est vraie (à savoir, la symétrie de l'information en espace polylogarithmique), alors VPSPACE n'a pas de circuits de taille polynomiale. Cette partie est assez différente du reste du manuscrit en ceci qu'elle traite exclusivement de complexité booléenne.

Le chapitre suivant est consacré à la preuve d'un autre résultat de transfert : si l'on souhaite séparer P de PAR sur les nombres complexes, alors il faut montrer que la classe VPSPACE n'est pas facilement calculable. En d'autres termes, sur $\mathbb{C}$, séparer P de PSPACE dans le modèle BSS implique la séparation de P et PSPACE dans le modèle de Valiant. L'ingrédient principal de la preuve sera les conditions de signe d'un ensemble de polynômes. On utilise pour cela un algorithme fonctionnant en espace polynomial pour énumérer les conditions de signe satisfaisables d'un ensemble de polynômes, ainsi que des suites d'entiers croissant suffisamment vite qui joueront le rôle de constantes transcendantes et permettront de tester l'appartenance d'un point à une variété.

Nous en sommes donc maintenant au sixième et dernier chapitre si l'on a bien compté. Celui-ci étend le résultat précédent au corps des réels. Pour cela, plus besoin de passer par des suites d'entiers croissant rapidement, car sur $\mathbb{R}$ on décide l'appartenance à une variété en testant si la somme des carrés des polynômes est nulle. En revanche, nous devons prendre en compte l'ordre sur les réels. Dans une première partie, on montre un résultat algorithmique indépendant : dans le corps à deux éléments, comment trouver efficacement un vecteur orthogonal à environ la moitié d'une collection de vecteurs. C'est un algorithme parallèle pour ce problème que l'on présente. Rien à voir avec cette thèse, pourrait-on objecter — si ce n'est pour justifier le P de LIP ? En réalité, nous aurons besoin de ce résultat dans la preuve du théorème de transfert. Nous

suivons en effet la construction de Grigoriev [30] d'un arbre de décision pour la localisation d'un point dans un arrangement d'hypersurfaces réelles, et cette dernière s'appuie sur un lemme non-constructif garantissant l'existence d'un vecteur orthogonal à environ la moitié d'une collection de vecteurs. Le résultat de Grigoriev ignore la complexité des polynômes mis en jeu et c'est pour la contrôler que nous avons montré une version constructive de ce lemme. Cela nous permet de généraliser le théorème de transfert du chapitre précédent au corps des réels.

Numérotation

Avant de commencer, nous devons un petit mot sur la numérotation. Tous les théorèmes, définitions, lemmes, propositions, remarques, etc., sont numérotés sous la forme du numéro de chapitre suivi d'une lettre (ou deux s'il y a eu plus de 26 items dans le chapitre) : on aura par exemple la remarque 1-A ou le théorème 3-AE. On utilise le même compteur pour tous, réinitialisé à chaque chapitre. Cela permet de localiser plus rapidement les résultats cités : par exemple, ledit théorème 3-AE se trouve au chapitre 3, après le lemme 3-AD et avant le corollaire 3-AF, comme il se doit.

CHAPITRE
UN

Notions et définitions

DANS ce chapitre, nous présentons les principales notions et définitions que nous allons utiliser tout au long de ce document. On verra les classes de complexité booléennes usuelles, les classes de complexité de Valiant et de Blum, Shub et Smale (BSS). Le lecteur risque quelqu'indigestion car il s'agit principalement d'un chapelet de définitions absconses mais pourtant classiques. La meilleure façon de lire ce chapitre est certainement de ne pas le lire ; mieux vaut s'y référer par la suite lorsqu'une notion imprévue surgit.

La question de l'uniformité des circuits est également abordée ; c'est certainement la partie la plus intéressante du chapitre. Et puis quand même, c'est le seul chapitre où il y a des dessins, il faut en profiter.

1.1 Classes booléennes, partie 1

Nous donnons ici un aperçu des classes que nous rencontrerons dans ce manuscrit. On pourra se référer au livre de Papadimtriou [63] ou à celui de Balcázar, Díaz et Gabarró [4] pour les plus connues d'entre elles, au panorama plus complet du livre d'Hemaspaandra et Ogiwara [33] pour les autres, voire au *Complexity Zoo* [22] pour les plus curieux.

Langages

Nous sommes ici dans le cas booléen : un langage est simplement un ensemble de mots sur l'alphabet $\{0,1\}$. Ainsi, un langage A est une partie de $\{0,1\}^*$. Si n est un entier, on notera $A^{=n}$ l'ensemble des mots de A de taille n.

Les classiques

Pour une présentation des machines de Turing, on choisira son livre favori de calculabilité ou de complexité. Pour plus de commodité, on travaillera avec des machines de Turing à plusieurs rubans, notamment pour pouvoir y adjoindre facilement un oracle

au paragraphe suivant. Pour fixer les idées, nos machines auront un ruban d'entrée (en lecture seule), deux rubans de travail et éventuellement un ruban d'oracle. Ce sont ces trois derniers rubans qui servent à compter l'espace utilisé par la machine.
Si $f : \mathbb{N} \to \mathbb{N}$ est une fonction, on note $\mathsf{DTIME}(f(n))$ (respectivement $\mathsf{DSPACE}(f(n))$) l'ensemble de langages A reconnus par une machine de Turing fonctionnant en temps $O(f(n))$ (resp. travaillant en espace $O(f(n))$). On note $\mathsf{NTIME}(f(n))$ l'ensemble de langages A reconnus par une machine de Turing non-déterministe en temps $O(f(n))$. On définit alors les classes suivantes :

L	$= \mathsf{DSPACE}(\log n)$	(espace logarithmique),
P	$= \cup_{k\geqslant 0}\mathsf{DTIME}(n^k)$	(temps polynomial),
NP	$= \cup_{k\geqslant 0}\mathsf{NTIME}(n^k)$	(temps non-déterministe polynomial),
PSPACE	$= \cup_{k\geqslant 0}\mathsf{DSPACE}(n^k)$	(espace polynomial),
EXP	$= \cup_{k\geqslant 0}\mathsf{DTIME}(2^{n^k})$	(temps exponentiel).

1-A – Remarque L'espace requis par une machine de Turing dans un calcul peut aussi être vu comme le temps mis par un algorithme parallèle. Il s'agit de la *parallel computation thesis* (voir par exemple Papadimitriou [63, p. 398]) : le temps parallèle $f(n)$ est inclus dans l'espace $f(n)$ qui est lui-même inclus dans le temps parallèle $f(n)^2$. On verra par la suite le lien avec la profondeur des circuits, facilement interprétée en termes de temps parallèle.

Oracles

Nous rappelons ici l'idée intuitive d'oracles. Un oracle A est un langage. Une machine $\mathscr{M}$ avec oracle A est simplement une machine de Turing à plusieurs rubans (dont un « ruban d'oracle ») munie d'un état spécial permettant d'interroger l'oracle. Lorsque la machine entre dans cet état, à l'étape d'après elle entre dans un nouvel état q_{oui} ou q_{non}, selon que le mot sur le ruban d'oracle de la machine est respectivement dans A ou non. Ainsi, en une seule étape de calcul, on sait si un mot est dans le langage A ou non, et ce quelle que soit la complexité de A.

On peut adjoindre un oracle à une classe de complexité : il suffit de munir de cet oracle les machines de Turing définissant la classe. On notera alors l'oracle en exposant de la classe. Par exemple, P^A est l'ensemble des langages reconnus en temps polynomial par des machines de Turing avec l'oracle A.

Enfin, si l'oracle peut être choisi arbitrairement dans toute une classe $\mathscr{C}$, on notera cette classe en exposant : par exemple, P^{NP} est la classe des langages reconnus en temps polynomial par des machines de Turing munies d'un oracle NP.

Hiérarchie polynomiale

La hiérarchie polynomiale, introduite par Stockmeyer [72], est constituée de niveaux notés Σ_i^{p} pour $i \geqslant 0$ (ici, p signifie polynomial pour différencier de la hiérarchie arithmétique). Ces niveaux sont définis comme des classes à oracles :

$$\Sigma_0^{\mathsf{p}} = \mathsf{P} \text{ et } \Sigma_{i+1}^{\mathsf{p}} = \mathsf{NP}^{\Sigma_i^{\mathsf{p}}}.$$

On peut aussi définir les classes Σ_i^{P} grâce à une alternance de i quantificateurs, on pourra voir à ce sujet Papadimitriou [63] par exemple. La réunion de toutes ces classes forme la hiérarchie polynomiale :

$$\mathsf{PH} = \bigcup_{i \geqslant 0} \Sigma_i^{\mathsf{P}}.$$

Classes probabilistes

Nous verrons deux classes d'algorithmes probabilistes : BPP et RP. La première est l'ensemble des problèmes résolus par des machines probabilistes pouvant faire des erreurs « de chaque côté » ; pour la seconde en revanche, seul un côté est autorisé. Formellement ça donne quelque chose comme ce qui suit. Un langage A est dans BPP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$\begin{array}{rcl} x \in A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (2/3)2^{p(|x|)} \text{ et} \\ x \notin A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \leqslant (1/3)2^{p(|x|)}. \end{array}$$

De la même manière, A est dans RP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$\begin{array}{rcl} x \in A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (1/2)2^{p(|x|)} \text{ et} \\ x \notin A & \Longrightarrow & \{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\} = \emptyset. \end{array}$$

Classes de comptage

Lorsque le seuil entre acceptation et rejet est nul dans l'algorithme probabiliste, on obtient la classe PP. On ne parle plus vraiment d'une classe probabiliste, il s'agit plutôt d'une classe de comptage. En voici la définition formelle. Un langage A est dans PP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (1/2)2^{p(|x|)}$$

Sous ses airs anodins, cette classe est assez volumineuse : Toda [75] a montré qu'utilisée en oracle d'une machine polynomiale, elle contient toute la hiérarchie polynomiale. En d'autres termes, le théorème de Toda s'écrit ainsi :

$$\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}.$$

De la même façon, on peut compter le nombre de mots y modulo un entier. On obtient ainsi les classes $\mathsf{Mod}_k\mathsf{P}$, dont voici la définition : si $k \geqslant 2$ est un entier, un langage A est dans $\mathsf{Mod}_k\mathsf{P}$ s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \not\equiv 0 \bmod k.$$

Lorsque k est un nombre premier, Beigel et Gill [5] ont montré que $\mathsf{Mod}_k\mathsf{P}$ est close par complément : on peut donc prendre $\equiv 0$ plutôt que $\not\equiv 0$ dans la définition. Comme le veut l'usage, on notera $\oplus\mathsf{P}$ la classe $\mathsf{Mod}_2\mathsf{P}$.

Hiérarchie de comptage

La hiérarchie de comptage introduite par Wagner [84] est définie à partir de la classe PP d'une manière similaire à la hiérarchie polynomiale à partir de NP. Ainsi, elle est constituée des niveaux C_i pour $i \geqslant 0$ définis comme suit :

$$\mathsf{C}_0 = \mathsf{P} \text{ et } \mathsf{C}_{i+1} = \mathsf{PP}^{\mathsf{C}_i}.$$

La hiérarchie de comptage est alors

$$\mathsf{CH} = \cup_{i \geqslant 0} \mathsf{C}_i.$$

Classes de fonctions

Des classes de *fonctions* sont également définies : il s'agit de fonctions $f : \{0,1\}^* \to \mathbb{N}$ qui prennent un mot booléen et renvoient un entier. Ainsi, une fonction f est dans #P s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}|.$$

La classe GapP est alors la clôture de #P par soustraction, c'est-à-dire que l'on considère maintenant des fonctions $g : \{0,1\}^* \to \mathbb{Z}$.

Notons qu'il est facile de voir que $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$.

Complémentaire

Pour une classe de complexité $\mathscr{C}$, on notera co$\mathscr{C}$ l'ensemble des langages ${}^cA$ où $A \in \mathscr{C}$, c'est-à-dire l'ensemble des complémentaires des langages de $\mathscr{C}$. On rencontrera notamment coNP et coRP.

Conseils

On peut également aider le calcul d'une machine par un conseil, qui devra être le même pour tous les mots de même taille. On parle alors de classe non-uniforme. Voici la définition d'une classe avec conseil, issue de Karp et Lipton [39].

1-B – Définition

> Soit $\mathscr{C}$ une classe de complexité et $a : \mathbb{N} \to \mathbb{N}$ une fonction. La classe $\mathscr{C}/a(n)$ est l'ensemble des langages A tels qu'il existe une fonction $c : \mathbb{N} \to \{0,1\}^*$ et un langage $B \in \mathscr{C}$ satisfaisant :
> - pour tout n, $|c(n)| \leqslant a(n)$;
> - pour tout mot x, $x \in A \iff (x, c(|x|)) \in B$.
>
> Si, plutôt qu'une seule fonction a on a une famille F de fonctions, alors $\mathscr{C}/F = \cup_{a \in F} \mathscr{C}/a$.

Dans ce cadre, on peut définir l'ensemble poly des fonctions polynomialement bornées. Ainsi, on obtient par exemple P/poly (temps et conseil polynomiaux), NP/poly (temps non-déterministe et conseil polynomiaux), PSPACE/poly (espace et conseil polynomiaux), EXP/poly (temps exponentiel et conseil polynomial), etc.

Notons enfin que l'on définira une dernière classe booléenne à la section 1.3, mais nous aurons besoin pour cela de circuits. Il s'agira de la classe NC et de ses variantes selon l'uniformité des circuits.

On trouvera dans la figure 1.1 un résumé graphique illustrant les inclusions connues de quelques classes importantes (tout bon livre de complexité contient au moins un dessin de la sorte, éventuellement plus fourni). Bien sûr, pour la plupart des inclusions on ne sait pas si elles sont strictes.

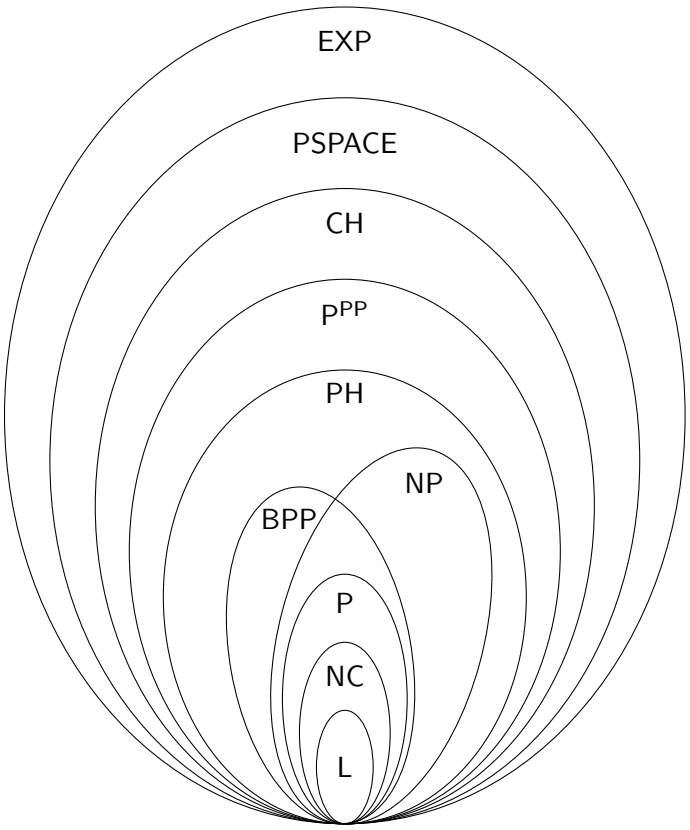


Fig. 1.1 – Morphologie des classes de complexité : du rachitisme à l'obésité.

1.2 Circuits

Les classes de complexité algébrique vont nécessiter l'introduction de circuits. Nous présentons plusieurs types de circuits : booléen, arithmétique et algébrique. Nous verrons dans la partie suivante que les classes booléennes peuvent également être définies en termes de circuits.

1-C – Définition

Un circuit est un graphe orienté sans cycle, dont les sommets, appelés portes, ont un degré entrant 0, 1 ou 2 et un degré sortant arbitraire. Une seule porte, appelée

sortie, a un degré sortant nul. Les portes de degré entrant nul sont appelées entrées ou constantes selon leur type. Les sommets du graphe sont étiquetés. Le choix pour les étiquettes dépend du type de circuit et du degré entrant de la porte.

On trouvera quelques illustrations dans les figures des paragraphes qui suivent.

Quelques caractéristiques des circuits

On définit deux caractéristiques essentielles des circuits : leur taille et leur profondeur.

1-D – Définition

Soit C un circuit. La taille de C, notée $|C|$, est le nombre de sommets du graphe. La profondeur de C est la taille du plus long chemin d'une entrée à la sortie.

En particulier, puisque chaque porte a un degré entrant $\leqslant 2$, la taille d'un circuit est majorée par 2^{p+1} où p est la profondeur du circuit.

Un circuit est simplement un objet booléen. Pour le décrire, il suffit de donner la liste des portes, leur type et leurs fils. On remarquera que cette description est de taille polynomiale en la taille du circuit.

Circuits booléens

Nous nous contentons des définitions ; on pourra se référer au livre de Vollmer [83] pour un traitement plus complet. Dans un circuit booléen, une porte est d'un des types suivants :

- une entrée, étiquetée par une variable x_i ;
- une porte NON, de degré entrant 1 et étiquetée par $\neg$;
- une porte ET (respectivement OU), de degré entrant 2 et étiquetée par $\wedge$ (resp. $\vee$).

Un exemple d'un tel circuit est donné à la figure 1.2.

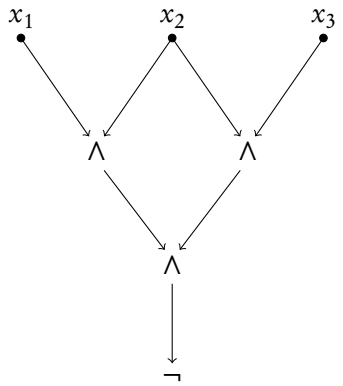


Fig. 1.2 – Un circuit booléen de taille 7 et de profondeur 3 valant 1 si le mot $x_1x_2x_3$ en entrée contient un zéro et 0 sinon.

Les entrées x_i prennent des valeurs booléennes. La valeur des autres portes est définie récursivement : c'est la négation de la valeur de son entrée pour une porte $\neg$ et la conjonc-

tion (respectivement disjonction) des valeurs de ses entrées pour une porte $\wedge$ (resp. $\vee$). La valeur du circuit est alors la valeur de la porte de sortie. Après avoir fixé la valeur des entrées, la valeur d'un circuit booléen est donc un élément de $\{0,1\}$.

Circuits arithmétiques

Sur les circuits arithmétiques, on pourra lire Malod [57] par exemple. Dans un circuit arithmétique, on se place sur un corps K quelconque. Une porte est alors d'un des types suivants :

- une constante, étiquetée par une constante $c \in K$;
- une entrée, étiquetée par une variable x_i ;
- une porte d'addition (respectivement de multiplication), de degré entrant 2 et étiquetée par $+$ (resp. par $\times$).

La figure 1.3 présente un exemple de circuit arithmétique.

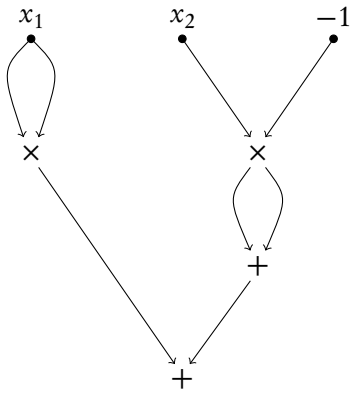


Fig. 1.3 – Un circuit arithmétique de taille 7 et de profondeur 3 calculant le polynôme $x_1^2 - 2x_2$.

Un tel circuit calcule un polynôme à coefficients dans K défini comme suit. Les portes constantes étiquetées par $c \in K$ calculent le polynôme constant c. Les entrées x_i calculent le polynôme x_i. Le polynôme calculé par les autres portes est défini récursivement : c'est la somme des polynômes entrant pour une porte $+$ et leur produit pour une porte $\times$. Le polynôme calculé par le circuit est alors celui calculé par la porte de sortie.

Un circuit arithmétique sur K calcule donc un polynôme à coefficients dans K et à plusieurs variables $x_1, \ldots, x_n$. Si la seule constante autorisée est -1, on parlera de circuit sans constante : le polynôme calculé aura alors des coefficients entiers. Afin de simplifier les preuves, nous avons fait le choix de considérer des circuits sans porte de soustraction, c'est pourquoi nous devons utiliser -1 comme constante ; nous aurions pu définir des circuits avec soustraction et dans ce cas on aurait plus naturellement choisi d'autoriser la seule constante 1 dans la version sans constante.

Les circuits arithmétiques sont au moins aussi puissants que les circuits booléens au sens suivant.

1-E – Lemme

Soit C un circuit booléen de taille t. Il existe un circuit arithmétique C' de taille $\leqslant 4t$ qui simule C dans le sens suivant : sur toute entrée booléenne $\bar{x} \in \{0,1\}^n$, on a $C'(\bar{x}) = C(\bar{x})$.
De plus, la construction de C' à partir de C se fait en espace logarithmique.

Preuve Il suffit de remplacer $x \wedge y$ par xy, $x \vee y$ par $x + y - xy$ et $\neg x$ par $1 - x$. ∎

Dans ce qui suit (notamment au chapitre 2), nous aurons besoin d'encoder en binaire des polynômes à coefficients entiers. Un monôme $\gamma x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, avec $\gamma \in \mathbb{Z}$ et $\alpha_i \in \mathbb{N}$, sera encodé par le uple $(\gamma, \alpha_1, \ldots, \alpha_n)$ où toutes ces composantes sont données en binaire. La *représentation dense* d'un polynôme à coefficients entiers (ou rationnels) sera alors la liste des encodages de tous ses monômes.

Circuits algébriques

La troisième catégorie de circuits permet de reconnaître des langages sur un corps K. Les circuits algébriques sont traités en détail dans le livre de Poizat [65]. Une porte est alors d'un des types suivants :

- une constante, étiquetée par une constante $c \in K$;
- une entrée, étiquetée par une variable x_i ;
- une porte de test d'égalité (respectivement d'inégalité si le corps K est ordonné), de degré entrant 1 et étiquetée par $=$ (resp. $\leqslant$) ;
- une porte d'addition (respectivement de multiplication), de degré entrant 2 et étiquetée par $+$ (resp. par $\times$).

On impose de plus que la porte de sortie soit une porte de test. On donne un exemple d'un tel circuit à la figure 1.4.

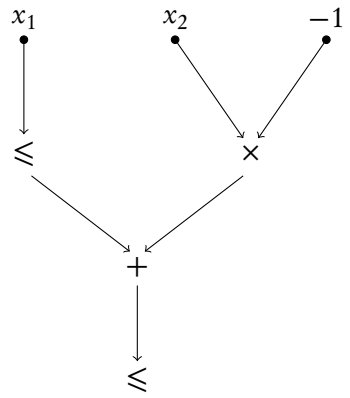


Fig. 1.4 – Un circuit algébrique de taille 7 et de profondeur 3 dont la valeur est $(x_1 \leqslant 0 \Rightarrow x_2 \geqslant 1) \wedge (x_1 > 0 \Rightarrow x_2 \geqslant 0)$.

La valeur d'une porte est encore définie par induction. Les variables x_i prennent une valeur dans K. La valeur d'une porte étiquetée par une constante $c \in K$ est c. La valeur d'une porte d'addition (respectivement de multiplication) est la somme (resp. le produit)

de ses entrées. La valeur d'une porte de test d'égalité (respectivement d'inégalité) est 1 si la valeur de son entrée est nulle (resp. $\leqslant 0$), 0 sinon. La valeur du circuit est la valeur de la porte de sortie. Ainsi, après avoir fixé la valeur des entrées, la valeur d'un circuit algébrique est donc un élément de $\{0,1\}$ puisque la porte de sortie est une porte de test.

Uniformité

Les circuits sont des modèles de calcul intrinsèquement non-uniformes, c'est-à-dire qu'il y a un dispositif différent pour chaque longueur de mots d'entrée. Comme on va le voir, on peut toutefois imposer à une famille de circuits d'être uniforme en passant par les machines de Turing (qui, elles, sont intrinsèquement uniformes). Cela donne lieu à plusieurs notions d'uniformité selon la puissance de la machine de Turing que l'on considère. Nous définissons tout cela ci-dessous et verrons plus loin que ces différentes notions sont parfois équivalentes.

1-F – Définition

Soient (C_n) une famille de circuits et $\mathscr{C}$ une classe de complexité. On considère le langage suivant (la description de (C_n))

$$\mathrm{DES}_C = \{(1^n, i, b) \mid \text{le } i\text{-ème bit de l'encodage de } C_n \text{ est } b\}.$$

On dit que la famille (C_n) est $\mathscr{C}$-uniforme si $\mathrm{DES}_C \in \mathscr{C}$.

Les classes usuelles pour $\mathscr{C}$ dans la définition ci-dessus sont L, P et PSPACE. Sauf mention contraire, lorsqu'on parlera d'uniformité dans la suite, il s'agira d'uniformité polynomiale (c'est-à-dire de P-uniformité). On remarquera que, dans certains cas et notamment pour PSPACE et ses versions algébriques, cela n'a guère d'importance puisque ces trois notions sont équivalentes comme nous le montrerons plus loin. Nous utiliserons brièvement aussi la P/poly-uniformité (ce qui n'est pertinent que lorsque le circuit a une taille superpolynomiale).

1-G – Remarque Lorsque le circuit est de taille polynomiale, il est équivalent de reconnaître le langage DES_C et de construire la description du circuit. En revanche, pour un circuit de taille exponentielle, le temps requis pour la construction du circuit sera toujours exponentiel alors qu'on pourrait reconnaître DES_C en temps polynomial.

1.3 Classes booléennes, partie 2

Les classes de complexité booléenne peuvent aussi être définies à l'aide de circuits booléens. En effet, il est bien connu, depuis Karp et Lipton [39] que P/poly est l'ensemble des langages reconnus par des circuits de taille polynomiale. Nous approfondissons ces liens dans cette partie : une condition d'uniformité sur les familles de circuits booléens permet de retrouver les classes de complexité uniformes. Comme nous l'avons dit, nous travaillerons principalement avec des classes uniformes, malgré quelques exceptions.

Langage reconnu par des circuits

Soient C un circuit booléen à n entrées et x un mot. On dit que x est accepté par C si $|x| = n$ et la valeur de C sur x, notée $C(x)$, est 1. Sinon, c'est-à-dire si $C(x) = 0$ ou $|x| \neq n$, on dit que x est rejeté par C.

Soit (C_n) une famille de circuits booléens, le circuit C_n ayant n entrées (c'est-à-dire qu'il accepte en entrée des mots de taille n). Le langage reconnu par la famille (C_n) est l'ensemble des mots reconnus par un des circuits C_n.

Temps, taille

On peut grossièrement assimiler la taille d'un circuit au temps de calcul d'un algorithme séquentiel. C'est la signification du lemme suivant. Pour la preuve, on introduit tout d'abord un langage P-complet.

1-H – Définition

Le langage PVC (*problème de la valeur d'un circuit*) est l'ensemble des circuits booléens dont les entrées sont instanciées par 0 ou 1 et dont la valeur est 1.

Ce problème est P-complet pour les réductions en espace logarithmique, voir Papadimitriou [63, théorème 8.1].

1-I – Lemme

Soit A un langage. Les trois assertions suivantes sont équivalentes.

1. Le langage A est dans P.
2. Le langage A est reconnu par une famille L-uniforme de circuits booléens de taille polynomiale.
3. Le langage A est reconnu par une famille P-uniforme de circuits booléens de taille polynomiale.

Preuve La partie la moins facile, (1) implique (2), est montrée dans Papadimitriou [63, théorème 11.5]. La preuve repose sur la P-complétude du problème PVC pour les réductions logspace : il suffit de montrer que ce problème et la réduction sont calculables par des circuits L-uniformes.

L'implication (2) $\Rightarrow$ (3) est triviale. Pour montrer (3) $\Rightarrow$ (1), il suffit de voir qu'en temps polynomial on peut construire le circuit P-uniforme reconnaissant A et le simuler. ∎

En autorisant (temporairement) plusieurs portes de sortie dans nos circuits booléens, on généralise aisément ce lemme aux fonctions calculables en temps polynomial. En particulier, le corollaire suivant nous sera utile dans la preuve du lemme 1-L.

1-J – Corollaire

Soit $f : \{0,1\}^* \to \{0,1\}^*$ une fonction calculable en temps polynomial telle que pour tous mots x, y, si $|x| = |y|$ alors $|f(x)| = |f(y)|$. Alors f est calculable par une famille L-uniforme de circuits de taille polynomiale.

Preuve Il suffit d'appliquer le lemme 1-I au langage suivant, reconnaissable en temps polynomial :

$$\{(x,i) \mid \text{le } i\text{-ème bit de } f(x) \text{ est } 1\}.$$

On obtient une famille L-uniforme de circuits de taille polynomiale. On calcule alors chaque bit de $f(x)$ en exécutant le circuit sur toutes les entrées (x,i). ∎

Espace, profondeur

On peut grossièrement assimiler la profondeur d'un circuit à l'espace utilisé par un calcul (ou encore au temps de calcul d'un algorithme parallèle, voir la remarque 1-A). Le lemme suivant précise cette idée. Par ailleurs, il est intéressant de constater que les trois notions d'uniformité L, P et PSPACE coïncident pour les circuits de profondeur polynomiale : la raison est que l'on a des problèmes décidés par des circuits de structure simple (mais éventuellement gros) qui sont PSPACE-complets pour les réductions logspace.

Pour la preuve, nous aurons besoin du langage suivant.

1-K – Définition

Le langage QBF (*quantified boolean formula*) est l'ensemble des formules vraies de la forme $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$, où ϕ est une formule booléenne du premier ordre sans quantificateur.

L'intérêt de ce langage est qu'il est PSPACE-complet pour les réductions logspace. On pourra voir pour cela le livre de Papadimitriou [63].

1-L – Lemme

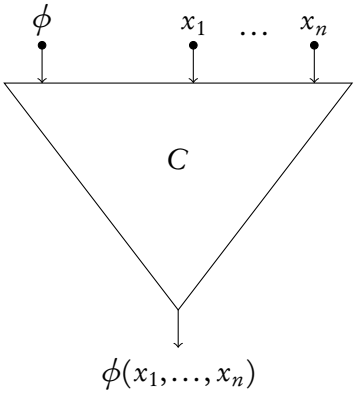
Soit A un langage. Les quatre assertions suivantes sont équivalentes.

1. Le langage A est dans PSPACE.
2. Le langage A est reconnu par une famille L-uniforme de circuits booléens de profondeur polynomiale.
3. Le langage A est reconnu par une famille P-uniforme de circuits booléens de profondeur polynomiale.
4. Le langage A est reconnu par une famille PSPACE-uniforme de circuits booléens de profondeur polynomiale.

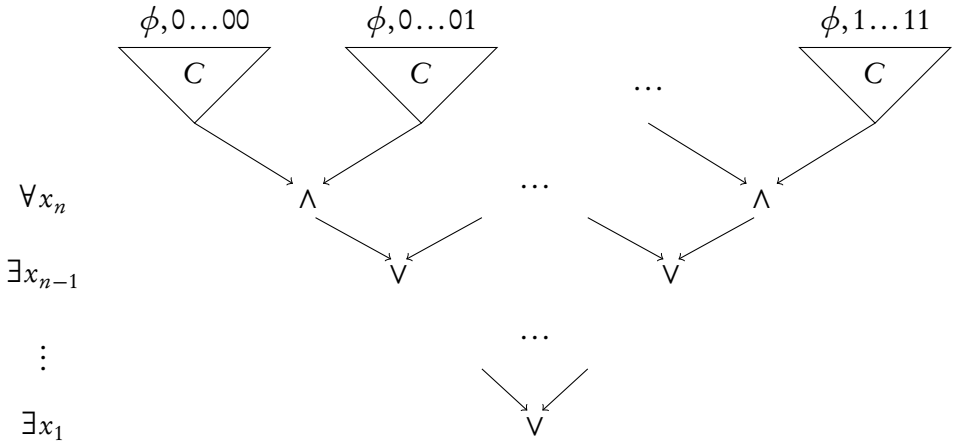
Preuve Voici le plan de la preuve : nous montrons qu'un langage $A \in$ PSPACE est reconnu par une famille logspace-uniforme de circuits booléens de profondeur polynomiale, donc *a fortiori* par une famille P-uniforme ou PSPACE-uniforme ; puis nous montrons qu'une famille PSPACE-uniforme peut être simulée par un algorithme fonctionnant en espace polynomial.

Pour la première partie, nous allons utiliser la PSPACE-complétude de QBF pour les réductions logspace. Soit f la réduction de A à QBF, c'est-à-dire que $x \in A \iff f(x) \in$ QBF. Remarquons en premier lieu que l'on peut supposer que pour tous mots x, y, si $|x| = |y|$ alors $|f(x)| = |f(y)|$: en effet, il suffit d'ajouter éventuellement des variables inutiles à nos instances $f(x)$ ou $f(y)$ de QBF. Ainsi, le corollaire 1-J

montre que f est calculable par une famille L-uniforme de circuits de taille polynomiale (donc en particulier de profondeur polynomiale). Pour décider A, il suffit maintenant de composer les circuits pour f avec des circuits pour QBF. Il suffit donc de fournir une famille L-uniforme de circuits pour le problème QBF. Quitte à ajouter des variables inutiles, on peut supposer que toute entrée de taille $2n$ a n variables. Ainsi, une entrée de taille $2n$ est de la forme $\exists x_1 \forall x_2 \exists x_3 \dots Q x_n \phi(x_1, \dots, x_n)$, où x_i prend ses valeurs dans $\{0, 1\}$. Enfin, on peut supposer que ϕ est donnée sous forme d'arbre (dont les sommets sont étiquetés par $\wedge$, $\vee$ et $\neg$).
Si les variables x_i sont fixées, alors on peut évaluer $\phi(x_1, \dots, x_n)$ simplement en parcourant l'arbre. Cela se fait par un circuit C logspace-uniforme comme sur la figure ci-dessous.



Maintenant, on peut construire le circuit entier, en partant de la sortie : d'abord on met un arbre binaire complet, de profondeur n, le i-ème niveau correspondant à la variable x_i. Les sommets du niveau i sont donc étiquetés par $\vee$ si le quantificateur correspondant est existentiel et par $\wedge$ s'il est universel. Puis on met au-dessus le circuit C pour évaluer $\phi(x_1, \dots, x_n)$, en remplaçant $x_1, \dots, x_n$ par les valeurs correspondant à leur emplacement. Tout cela est illustré sur la figure ci-dessous. On se convainc aisément que cette construction se fait en espace logarithmique.



Il reste maintenant le second point du lemme, à savoir, montrer qu'en espace polyno-

mial on peut évaluer une famille PSPACE-uniforme de circuits booléens de profondeur polynomiale. Encore une fois, on part de la sortie : il suffit d'évaluer tour à tour les deux sous-circuits de cette porte (ou l'unique sous-circuit s'il s'agit d'une porte ¬). On connaît cette porte ainsi que ses fils par la description PSPACE du circuit. Pour évaluer la valeur de la porte, on le fait par induction : on évalue d'abord le fils gauche, on garde le résultat en effaçant l'espace de calcul, puis le fils droit. On efface l'espace de calcul une nouvelle fois et on effectue l'opération sur les deux résultats. L'espace utilisé est donc une constante plus le max de l'espace utilisé pour le fils droit et pour le fils gauche. En tout, on a utilisé un espace polynomial car la profondeur est polynomiale, ce qui prouve le lemme. ∎

Il est facile de modifier la preuve précédente pour obtenir le résultat suivant.

1-M – Lemme

Un langage A est dans PSPACE/poly si et seulement s'il est reconnu par une famille P/poly-uniforme de circuits booléens de profondeur polynomiale.

Une dernière classe

Nous définissons enfin la dernière classe booléenne que nous utiliserons, NC. C'est la réunion des différentes classes NC^i pour $i \geqslant 1$. Nous présentons d'abord la version non-uniforme ; bien qu'il s'agisse d'une classe définie par des circuits et non par des conseils, pour souligner sa non-uniformité nous utiliserons la notation NC/poly.

Pour $i \geqslant 1$, la classe $\mathsf{NC}^i/\mathsf{poly}$ est l'ensemble des langages reconnus par des circuits booléens de taille polynomiale et de profondeur $O(\log^i n)$. La classe NC/poly est alors $\cup_{i\geqslant 1}\mathsf{NC}^i/\mathsf{poly}$, ou en d'autres termes, l'ensemble des langages reconnus par des circuits booléens de taille polynomiale et de profondeur polylogarithmique.

On définit aussi des versions uniformes : NC est la version L-uniforme de NC/poly et P-uniform NC est la version P-uniforme, comme on l'aura deviné. On ne sait pas si ces deux versions coïncident.

1.4 Modèle de Valiant

Dans le modèle de Valiant, on calcule des polynômes. Les éléments des classes de complexité sont des familles de polynômes (qui remplacent en quelque sorte les langages des classes booléennes). Nous donnons ici la définition des classes principales que nous utiliserons par la suite ; on pourra se référer au livre de Bürgisser [13] ou à la thèse de Malod [57] pour approfondir le sujet. D'autres classes plus spécifiques, comme VIIP ou VPSPACE, seront définies dans les chapitres concernés.

Calcul de polynômes

On s'intéresse à des suites de polynômes, calculées par des suites de circuits arithmétiques. On utilisera indifféremment l'expression « famille de polynômes » (ou de circuits), et on notera (f_n), pour signifier une suite $(f_n)_{n\in\mathbb{N}}$ de polynômes (ou de circuits).

1-N – Définition

Soient K un corps quelconque, (f_n) une famille de polynômes à plusieurs variables, $f_n \in K[x_1, \ldots, x_{u(n)}]$, et (C_n) une famille de circuits arithmétiques dont les constantes éventuelles sont dans K. On dit que la famille de polynômes (f_n) est calculée par la famille de circuits (C_n) si pour tout n, le circuit C_n calcule le polynôme f_n.

Classes de Valiant

Les deux classes originales, VP et VNP, sont définies dans Valiant [79]. Ce sont des classes de familles non-uniformes de polynômes, de degré polynomialement borné et utilisant des constantes arbitraires.

En essence, la classe VP est l'ensemble des familles de polynômes calculables par des circuits de taille polynomiale (« facilement calculables »), et VNP consiste en une somme exponentielle de familles VP. Une définition plus précise est donnée ci-dessous. La plupart du temps, nous abrégerons $x_1, \ldots, x_{u(n)}$ en $\bar{x}$.

1-O – Définition

Soit K un corps quelconque.

- La classe VP est l'ensemble des familles de polynômes (f_n) de degré polynomialement borné, $f_n \in K[x_1, \ldots, x_{u(n)}]$, calculées par une famille de circuits de taille polynomiale.
- Une famille $(f_n(\bar{x}))$ est dans la classe VNP s'il existe une famille $(g_n(\bar{x}, \bar{y})) \in \mathsf{VP}$ telle que
$$f_n(\bar{x}) = \sum_{\bar{\epsilon} \in \{0,1\}^{|\bar{y}|}} g_n(\bar{x}, \bar{\epsilon}).$$

Évidemment, la question « $\mathsf{VP} = \mathsf{VNP}$? » est ouverte ; pour reprendre la terminologie de Bürgisser [13], nous désignerons par « hypothèse de Valiant » la conjecture que VNP diffère de VP. Avant de voir les variantes annoncées de ces classes, nous introduisons la notion de VNP-complétude (Valiant [79]). Nous avons d'abord besoin d'une notion de réduction.

1-P – Définition

Soit K un corps quelconque.

- Un polynôme g est une *projection* d'un polynôme f s'il existe $a_1, \ldots, a_m$ tels que $g(x_1, \ldots, x_n) = f(a_1, \ldots, a_m)$, où les a_i sont des éléments de K ou des variables parmi $x_1, \ldots, x_n$.
- Soient (f_n) et (g_n) deux familles de polynômes. On dit que (g_n) est une p-projection de (f_n) s'il existe une fonction $t(n)$ polynomialement bornée telle que pour tout n, g_n soit une projection de $f_{t(n)}$.
- Soit (f_n) une famille de polynômes. On dit que (f_n) est VNP-complète si $(f_n) \in \mathsf{VNP}$ et si toute famille $(g_n) \in \mathsf{VNP}$ est une p-projection de (f_n).

Un des résultats célèbres de Valiant est la VNP-complétude du permanent sur tout corps

de caractéristique différente de 2. Le permanent est la famille (per_n) définie comme suit :

$$\mathrm{per}_n(x_{1,1},\ldots,x_{1,n},x_{2,1},\ldots,x_{n,n}) = \sum_{\sigma\in\mathscr{S}_n}\prod_{i=1}^{n} x_{i,\sigma(i)},$$

où la somme est prise sur toutes les permutations σ de $\{1,\ldots,n\}$. Malgré la similarité avec le déterminant (la différence étant seulement que la signature de la permutation est prise en compte dans ce dernier), ce résultat de complétude, avec celui de #P-complétude dans un cadre booléen, semble indiquer que le permanent est difficile à calculer. Une autre famille VNP-complète, sur tout corps K cette fois, est le hamiltonien, où la somme est prise seulement sur les n-cycles σ. Nous retrouverons ce polynôme aux chapitres 2 et 4.

Classes de Valiant-Malod

Dans ce paragraphe et le suivant, nous présentons plusieurs variantes des classes de Valiant, selon l'uniformité, le degré et l'utilisation de constantes. Dans un cadre non-uniforme, ces variantes sont issues de Malod [57]. Dans le reste de ce manuscrit, nous nous concentrerons essentiellement sur le cas des familles uniformes et sans constante.

Malod [57] a introduit une version de VP et VNP où le degré des polynômes n'est plus restreint : il peut être exponentiel, la seule contrainte étant la taille polynomiale du circuit. Il s'agit donc exactement de la définition 1-O sans la contrainte sur le degré. On notera ces classes $\mathsf{VP}_{\mathsf{nb}}$ et $\mathsf{VNP}_{\mathsf{nb}}$, où nb signifie *non borné*.

Des deux classes en degré non borné, on définit aisément des versions sans constante, en imposant que l'unique constante des circuits soit -1. On obtient ainsi les classes $\mathsf{VP}^0_{\mathsf{nb}}$ et $\mathsf{VNP}^0_{\mathsf{nb}}$. En revanche, les versions sans constante des classes de degré borné sont un peu plus délicates à définir afin d'éviter que des constantes de taille exponentielle ne puissent être calculées. Elles reposent sur la notion de degré formel complet d'un circuit de Malod [57] (aussi dans Miller, Ramachandran et Kaltofen [61]), que l'on définit maintenant.

1-Q – Définition

Soit C un circuit arithmétique. Le *degré formel complet* d'une porte de C est défini par induction : le degré formel complet d'une constante ou d'une variable est 1 ; celui d'une porte $+$ est le max des degrés de ses entrées ; celui d'une porte $\times$ est la somme des degrés de ses entrées.
Le degré formel complet de C, noté $\deg_{\mathrm{f}}(C)$, est alors le degré formel complet de la porte de sortie de C.

Voici donc la définition des versions sans constante de VP et VNP.

1-R – Définition

- Une famille de polynômes (f_n) appartient à la classe VP^0 si (f_n) est calculé par une famille de circuits arithmétiques sans constante de taille et de degré formel complet polynomialement bornés.
- Une famille $(f_n(\bar{x}))$ est dans la classe VNP^0 s'il existe une famille $(g_n(\bar{x},\bar{y}))\in$

VP^0 telle que

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

Uniformité

Des quatre classes sans constante, on définit aisément des versions uniforme, en imposant une condition d'uniformité polynomiale sur les circuits. On obtient ainsi les classes Uniform VP^0, Uniform VNP^0, Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ et Uniform $\mathsf{VNP}^0_{\mathsf{nb}}$.

1-S – Remarque Dans les classes avec constantes, une notion d'uniformité a moins de sens car les constantes peuvent être arbitraires, c'est pourquoi nous ne définirons pas de variantes uniformes des classes avec constantes.

Le lemme suivant montre que considérer l'uniformité L ou polynomiale ne change rien pour les classes en degré non borné. La preuve repose sur la P-complétude pour la réduction logspace d'un problème très uniforme.

1-T – Lemme

Les classes Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ et Uniform $\mathsf{VNP}^0_{\mathsf{nb}}$ sont définies de manière équivalente en considérant l'uniformité P ou l'uniformité L.

Preuve Par définition de $\mathsf{VNP}_{\mathsf{nb}}$, il suffit de montrer le résultat pour $\mathsf{VP}_{\mathsf{nb}}$.

Soit (C_n) une famille P-uniforme de circuits de taille polynomiale, calculant une famille de polynômes (f_n). On veut construire une famille L-uniforme de circuits (D_n) de taille polynomiale calculant les mêmes polynômes. L'idée est la suivante : D_n construit C_n au fur et à mesure en utilisant sa description, tout en le simulant. Afin de rester L-uniforme, on utilisera le lemme 1-I.

Puisque la description de (C_n) est dans P, on peut appliquer le lemme 1-I : soit (B_n) la famille L-uniforme de circuits booléens de taille polynomiale qui décrit (C_n). Ainsi, le circuit B_n nous dit le type t de la porte i de C_n (étiquetée par $\circ$) : par exemple $t = 0$ si $\circ = +$ et $t = 1$ si $\circ = \times$. Pour calculer $x \circ y$, il suffit alors de calculer $txy + (1-t)(x+y)$. Or par le lemme 1-E, on peut simuler le circuit booléen B_n par un circuit arithmétique B'_n en gardant l'uniformité logspace.

Ainsi, on remplace chaque porte de C_n par une copie de B'_n avec pour entrée le code binaire correspondant au numéro de la porte ; à la sortie de B'_n on a le type de la porte et on applique ce qui précède pour effectuer le bon calcul de C_n. ∎

1-U – Remarque En revanche, la preuve ne fonctionne plus pour les version en degré borné, car on ne peut alors pas simuler n'importe quel circuit booléen de taille polynomiale à cause de la contrainte du degré. Il serait intéressant de savoir si l'uniformité polynomiale est strictement plus puissante que l'uniformité logspace pour VP^0 et VNP^0.

1.5 Modèle BSS

Le modèle de Blum, Shub et Smale (BSS) est un autre modèle de calcul algébrique, mais cette fois on reconnaît des langages (comme dans le cas booléen) plutôt que de

calculer des polynômes. Ce modèle a été introduit dans [9] par Blum, Shub et Smale, d'où son nom, une dizaine d'années après celui de Valiant. On pourra se référer au livre de Blum, Cucker, Shub et Smale [8] pour approfondir le sujet. Dans ce livre, le calcul sur $\mathbb{R}$ et $\mathbb{C}$ est introduit en utilisant une machine de Turing algébrique. Nous suivrons plutôt l'exposition du livre de Poizat [65], où le calcul sur une structure arbitraire est défini à l'aide de circuits algébriques. Nous nous contenterons de définir les classes BSS sur des corps ; nous fixons donc un corps K pour la suite.

Langages

Pour les langages sur K, l'alphabet est K et non $\{0,1\}$ comme pour les langages booléens. Un mot x de taille n sur K est un n-uplet d'éléments de K, c'est-à-dire un élément de K^n. Un langage A sur K est alors un ensemble de mots sur K, c'est-à-dire un sous-ensemble de $\cup_{n\geqslant 0} K^n$.

Si K n'est pas ordonné (par exemple $K=\mathbb{C}$), nous utiliserons l'abréviation K pour la structure $(K,+,\times,=)$; si K est ordonné (par exemple $K=\mathbb{R}$), l'abréviation K désignera la structure $(K,+,\times,\leqslant)$.

Reconnaissance de langages

Nous allons reconnaître des langages grâce à des circuits algébriques. Si K est ordonné, nous travaillerons avec des circuits algébriques munis de portes $+$, $\times$ et $\leqslant$; sinon, ce sera $+$, $\times$ et $=$. Nous avons imposé dans la définition que la valeur d'un circuit algébrique soit 0 ou 1 (car la porte de sortie est une porte de test) ; ainsi, comme pour un circuit booléen, on dit qu'un mot x est accepté par le circuit C si $C(x)=1$.

Le langage reconnu par une famille (C_n) de circuits algébriques, où C_n a n entrées, est l'ensemble des mots acceptés par ces circuits.

Classes usuelles

La gestion des constantes pour les classes BSS n'est pas la même que pour les classes de Valiant. Mais nous commençons par définir une classe sans constante qui, de ce fait, ne pose pas de problème. La classe P^0_K est l'ensemble des langages reconnus par une famille uniforme de circuits algébriques sans constante de taille polynomiale.

Pour la classe P_K, les constantes de nos circuits jouent le rôle de celles des machines de Blum, Shub et Smale [9] (voir aussi [8]). Ainsi, elles doivent être les mêmes pour tous les circuits de la famille. Formellement, un langage A est dans P_K s'il existe une famille uniforme $(C_n(\bar{x},\bar{y}))$ de circuits algébriques de taille polynomiale et un uple de constantes $\bar{a}$ de K tels que pour tout mot $\bar{x}\in K^n$ on ait

$$\bar{x}\in A \iff C_n(\bar{x},\bar{a})=1.$$

Comme dans le cas booléen, NP est la version existentielle de P. Ainsi, un langage $A\subseteq\cup_{n\geqslant 0}K^n$ est dans NP_K s'il existe un langage $B\in\mathsf{P}_K$ et un polynôme $p(n)$ tels que, pour tout mot x

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

La version sans constante NP^0_K est définie de même en prenant $B\in\mathsf{P}^0_K$.

Enfin, une sorte d'équivalent de PSPACE dans le modèle BSS est la classe PAR_K. Il s'agit des langages reconnus par une famille uniforme de circuits algébriques de profondeur polynomiale, avec la même convention que les constantes sont les mêmes pour toute la famille. On notera PAR_K^0 la version sans constante. Dans [18], cette classe est définie avec l'uniformité PSPACE. Mais, une nouvelle fois, les uniformités L, P et PSPACE sont équivalentes pour PAR_K^0. La preuve est similaire à celle du lemme 1-T mais en utilisant un langage PSPACE-complet pour les réduction logspace, par exemple QBF, c'est-à-dire qu'on utilise dans la preuve le lemme 1-L plutôt que 1-I.

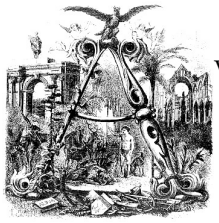
1-V – Lemme

> Soit K un corps. Les uniformités L, P et PSPACE sont équivalentes pour la classe PAR_K^0.

Voilà, si vous avez réussi à lire ce chapitre jusqu'ici, vous êtes prêt pour le reste du manuscrit (rassurez-vous, la réciproque n'est pas nécessairement vraie).

CHAPITRE
DEUX

Manipulation de circuits arithmétiques



AVANT d'aborder les notions de complexité algébrique proprement dite, nous étudions si l'encodage de polynômes par des circuits arithmétiques peut être manipulé efficacement. La raison pour laquelle nous nous intéressons à manipuler des circuits arithmétiques est qu'ils peuvent encoder des polynômes de manière très concise. L'exemple typique est le polynôme $(x+y)^{2^n}$ qui a un nombre de monômes exponentiel en n (donc une représentation de taille exponentielle en n dans l'encodage usuel par liste des monômes) mais est calculé par un circuit arithmétique de taille linéaire en n par élévations au carré successives.

Cette façon d'encoder les polynômes semble donc très intéressante. Malheureusement, nous allons montrer que leur manipulation est rendue difficile par ce codage compact. Nous nous intéressons pour cela à deux problèmes : décider si le coefficient d'un monôme est nul d'une part, calculer le degré du polynôme d'autre part. Nous introduisons également des variantes : partie de l'entrée en unaire (ce qui permet de contrôler le degré formel complet du circuit en fonction de la taille de l'entrée), arithmétique modulaire.

Travaux précédents. La manipulation de polynômes donnés par des circuits a été abondamment étudiée. Il existe bien sûr des algorithmes efficaces pour certains problèmes : citons, par exemple, un algorithme randomisé polynomial pour tester si deux polynômes sont égaux (Schwartz [69]). Certains autres problèmes ont des algorithmes efficaces lorsque le degré du polynôme est petit. C'est le cas par exemple du calcul du pgcd de deux polynômes (Kaltofen [37]). Enfin, plus récemment, Allender, Bürgisser, Kjeldgaard-Pedersen et Miltersen [2] étudient plusieurs problèmes, notamment calculer le degré d'un polynôme ou savoir si un entier donné par un circuit est positif. Ces problèmes sont dans la hiérarchie de comptage et semblent difficiles, au moins pour le dernier.

Plan du chapitre. Nous commençons par quelques notions utiles et rappels. Puis dans la section 2.2 nous définissons toutes les variantes des problèmes que nous allons étudier. La partie 2.3 étudie le problème du coefficient de monôme nul sur un corps de caractéristique nulle, alors que la partie suivante étudie sa variante en arithmétique modulaire. La

section 2.5 s'occupe du calcul du degré d'un polynôme. Enfin, on récapitule les résultat sous la forme d'un tableau dans la partie 2.6.

Ce chapitre, qui correspond à l'article [45], réutilise beaucoup de résultats de la thèse de Malod [57] mais le point de vue est celui de la complexité booléenne.

2.1 Notions utiles

2.1.1 Encodage des entrées

Rappelons qu'un monôme $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ est encodé par le uple $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ où les α_i sont des entiers donnés en binaire.

En ce qui concerne l'encodage des circuits arithmétiques, il s'agit de donner toutes les portes du circuit avec leurs entrées. On fera ainsi une liste de triplets : le i-ème élément de la liste est le triplet (t, j, k) correspondant à la porte i, où t est le type de porte (variable, constante -1, addition ou multiplication), j le numéro de la première entrée de la porte (s'il y en a) et k celui de la seconde.

2.1.2 Réduction

La notion de réduction Turing non-déterministe, notée $\leqslant_T^{sn}$ (pour *strong nondeterministic Turing reduction*), est une variante assez naturelle de la réduction à oracle $\leqslant_T$ classique : si A et B sont deux langages, $A \leqslant_T^{sn} B$ si $A \in \mathsf{NP}^B \cap \mathsf{coNP}^B$. Cette réduction a été introduite par Long [53].

2.1.3 Rappels

Rappelons que $\deg_{\mathrm{f}}(C)$ désigne le degré formel complet du circuit C (définition 1-Q). On notera $\deg(C)$ le degré du polynôme calculé par C. Lorsque m est un monôme, $\mathrm{coef}_C(m)$ est le coefficient du monôme m dans le polynôme calculé par C.

Dans ce chapitre, nous jonglerons avec un grand nombre de classes de complexité booléenne, voir pour cela la partie 1.1. En particulier, nous utiliserons des classes de comptage, par exemple PP, $\mathsf{Mod}_k\mathsf{P}$ ou encore CH.

Enfin, rappelons que le permanent d'une matrice carrée $M = (m_{i,j})_{1 \leqslant i,j \leqslant n}$ de taille $n \times n$ est défini comme suit :

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

2.2 Définition des problèmes

Le problème CMN, pour « coefficient de monôme nul », consiste à décider si le coefficient d'un monôme est nul :

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

Nous définissons également une version modulo k :

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

Si k est un nombre premier, cette version correspond au cas des corps de caractéristique k non nulle. Enfin, pour ces deux versions on définit également une « version unaire » où une partie de l'entrée est donnée en unaire. Plus précisément :

$$\mathrm{CMN}_{\mathrm{u}} = \{(C, m, 1^d) \mid \mathrm{coef}_C(m) = 0 \text{ et } \deg_{\mathrm{f}}(C) = d\} \text{ et}$$

$$\mathrm{CMN}_{\mathrm{u}}^k = \{(C, m, 1^d) \mid \mathrm{coef}_C(m) \equiv 0 \bmod k \text{ et } \deg_{\mathrm{f}}(C) = d\}.$$

Le second problème est le calcul du degré d'un polynôme. Plus précisément :

$$\mathrm{DEG} = \{(C, d) \mid \deg(C) \leqslant d\},$$

et on définit de même DEG^k pour des corps de caractéristique k. Il existe aussi une version « unaire » où d est donné en unaire :

$$\mathrm{DEG}_{\mathrm{u}} = \{(C, 1^d) \mid \deg(C) \leqslant d\},$$

ainsi que $\mathrm{DEG}_{\mathrm{u}}^k$ modulo k. Nous verrons qu'il y a pour cette version unaire un algorithme probabiliste polynomial, en d'autres termes que $\mathrm{DEG}_{\mathrm{u}} \in \mathsf{coRP}$.

Nous nous penchons tout d'abord sur le problème du coefficient de monôme nul. Nous verrons que ce problème est difficile, même en version unaire, mais qu'il est quand même dans la hiérarchie de comptage. Cela nous permettra ensuite de placer le calcul du degré dans la hiérarchie de comptage également et même un peu mieux pour la version modulo k.

2.3 Coefficient d'un monôme en caractéristique nulle

Dans cette partie, on étudie les problèmes CMN et $\mathrm{CMN}_{\mathrm{u}}$. Nous caractérisons la complexité du problème $\mathrm{CMN}_{\mathrm{u}}$ pour les réductions non-déterministes, ce qui implique une borne inférieure pour CMN. Nous donnons également une borne supérieure dans la hiérarchie de comptage pour CMN. Ces résultats sont résumés dans les deux propositions suivantes.

2-A – Proposition

> $\mathrm{CMN}_{\mathrm{u}}$ est $\mathsf{P}^{\#\mathsf{P}}$-complet pour les réductions Turing non-déterminisites $\leqslant_T^{sn}$.

2-B – Remarque Allender nous indique que le problème $\mathrm{CMN}_{\mathrm{u}}$ est $\mathsf{C}_{=}\mathsf{P}$-complet pour les réductions many-one. La preuve est similaire à celle donnée ci-dessous (voire plus simple). La classe $\mathsf{C}_{=}\mathsf{P}$, définie dans [84], consiste en l'ensemble des langages tels qu'il existe une machine de Turing non-déterministe fonctionnant en temps polynomial dont le nombre de chemins acceptants sur l'entrée x est calculable en temps déterministe polynomial. On pourra lire par exemple Toran [78] pour plus de détails sur cette classe.

2-C – Proposition

> CMN est dans la hiérarchie de comptage CH et est $\mathsf{P}^{\#\mathsf{P}}$-difficile pour les réductions Turing non-déterminisites $\leqslant_T^{sn}$.

Le reste de cette section est dévolu à la preuve de ces deux propositions. Nous montrons d'abord les bornes inférieures puis les bornes supérieures.

2.3.1 Borne inférieure

Le permanent

Puisque nous travaillons avec des langages et non des fonctions, on utilisera un langage capturant la complexité du calcul du permanent. Il s'agit du langage suivant (qui apparaît aussi dans Kabanets et Impagliazzo [36]) :

$$\mathrm{Perm} = \{(M, v) \mid \mathrm{per}(M) = v\},$$

où M est une matrice carrée et v un entier. Le langage Perm ainsi défini est difficile pour $\mathsf{P}^{\#\mathsf{P}}$ pour les réductions Turing non-déterministes, comme nous le prouvons maintenant. Cela vient du fait que le permanent est #P-complet (Valiant [80]).

Il s'agit donc de montrer que

$$\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathrm{Perm}} \cap \mathsf{coNP}^{\mathrm{Perm}}.$$

Prenons un langage $L \in \mathsf{P}^{\#\mathsf{P}}$ et montrons d'abord que $L \in \mathsf{NP}^{\mathrm{Perm}}$. Par complétude du permanent des matrices 0-1, il existe une machine de Turing polynomiale $\mathscr{M}$ avec l'oracle per qui décide L. Il est dès lors facile de construire une machine de Turing non-déterministe polynomiale $\mathscr{N}$ avec oracle Perm qui décide L. En effet, il suffit de simuler $\mathscr{M}$ de manière déterministe tant qu'elle ne fait pas appel à son oracle per. Lors d'une requête à l'oracle, la machine non-déterministe $\mathscr{N}$ devine la valeur du permanent demandé et vérifie qu'elle ne s'est pas trompé grâce à son oracle Perm. Si elle s'est trompé, on rejette (i.e. le chemin en cours est rejetant), sinon on connaît la valeur du permanent et on peut poursuivre la simulation de $\mathscr{M}$. Ainsi, $\mathscr{N}$ a exactement un chemin acceptant, et il donne la même réponse que $\mathscr{M}$, c'est-à-dire que $\mathscr{N}$ reconnaît L.

Afin de pouvoir effectuer la procédure ci-dessus en temps non-déterministe polynomial, il suffit simplement de s'assurer que le nombre de valeurs possibles prises par le permanent est simplement exponentiel. C'est immédiat puisque le permanent d'une matrice 0-1 carrée de taille n est inférieur à $n!$.

La clôture de $\mathsf{P}^{\#\mathsf{P}}$ par complément prouve la seconde inclusion $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{coNP}^{\mathrm{Perm}}$.

2-D – Remarque Le théorème de Toda [75] montre que $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$. Ainsi, par ce qui précède, on a

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

Cette inclusion implique que le langage Perm n'est pas dans la hiérarchie polynomiale si elle est stricte : en effet, si $\mathrm{Perm} \in \Sigma_i^p$ pour un certain $i \geqslant 0$, alors $\mathsf{PH} \subseteq \mathsf{NP}^{\Sigma_i^p} = \Sigma_{i+1}^p$, donc la hiérarchie polynomiale s'effondrerait au niveau $i+1$.

Nous avons donc un langage Perm « difficile »… Il reste à montrer que $\mathrm{CMN}_{\mathrm{u}}$ est plus dur que Perm. On retrouve les idées qui suivent dans Valiant [79].

Le permanent comme coefficient de monôme

Il s'agit maintenant de montrer que l'on peut décider Perm en temps polynomial dès lors qu'on sait décider $\mathrm{CMN}_{\mathrm{u}}$. Soient M une matrice carrée de taille n et v un entier. On utilise pour cela le très classique polynôme à plusieurs variables suivant (qui d'après

Malod [57] remonte au moins à Hammond en 1879 si l'on ne tient pas compte du terme $-vX_1X_2\cdots X_n$) :

$$P_{(M,v)}(X_1,\ldots,X_n)=\prod_{j=1}^{n}\Big(\sum_{i=1}^{n}m_{i,j}X_i\Big)-vX_1X_2\cdots X_n.$$

Le coefficient du monôme $X_1X_2\cdots X_n$ dans ce polynôme est $\mathrm{per}(M)-v$. En effet, lorsqu'on développe le produit de n termes afin d'obtenir le coefficient de $X_1X_2\cdots X_n$, il faut prendre dans chaque terme exactement une variable. Chacun de ces choix correspond à une permutation $\sigma\in S_n$: au terme numéro j du produit on associe la i-ème variable. La somme de tous ces choix donne donc $\sum_{\sigma\in S_n}\prod_{i=1}^{n}m_{i,\sigma(i)}$, soit $\mathrm{per}(M)$. Enfin, on retranche v pour obtenir le coefficient de $X_1X_2\cdots X_n$.

À partir du couple (M,v), où M est une matrice 0-1 carrée de taille n et v un entier positif inférieur à $n!$, il est facile de construire un circuit arithmétique calculant le polynôme $P_{(M,v)}$, de degré formel complet $O(n\log n)$. On procède ainsi : dans un premier temps, pour $1\leqslant i,j\leqslant n$, on calcule tous les $m_{i,j}X_i$, pour tout j on fait la somme sur i, puis on fait le produit sur j de tous ces termes. Le degré formel complet de cette construction est $2n$. Dans un deuxième temps, on calcule l'entier v à partir de la constante 1 et on multiplie par $X_1X_2\cdots X_n$. Cela requiert un degré formel complet inférieur à $n+2n\log n$ puisque $v\leqslant n!$. Enfin, on retranche $vX_1X_2\cdots X_n$ au produit précédent $\prod_{j=1}^{n}\Big(\sum_{i=1}^{n}m_{i,j}X_i\Big)$.

On note $C_{(M,v)}$ le circuit ainsi obtenu et d son degré formel complet, que l'on peut aisément calculer. Maintenant, pour savoir si $\mathrm{per}(M)=v$, il suffit de décider si le coefficient du monôme $X_1X_2\cdots X_n$ dans $P_{(M,v)}$ est nul ou, en d'autres termes, de décider si $(C_{(M,v)},X_1X_2\cdots X_n,1^d)\in\mathrm{CMN_u}$. Ainsi, pour les réductions many-one, $\mathrm{Perm}\leqslant\mathrm{CMN_u}$, ce qui montre que $\mathrm{CMN_u}$, et *a fortiori* CMN, est difficile pour $\mathsf{P}^{\#\mathsf{P}}$ pour les réductions non-déterministes à oracle. C'est la première partie des propositions 2-A et 2-C.

Notons au passage que si la hiérarchie polynomiale est stricte, alors CMN et $\mathrm{CMN_u}$ ne sont pas dedans, comme le montre la remarque 2-D.

2.3.2 Bornes supérieures

Nous montrons maintenant que la version unaire $\mathrm{CMN_u}$ est dans $\mathsf{P}^{\#\mathsf{P}}$ et que la version générale CMN est dans la hiérarchie de comptage. Pour le premier, nous utilisons les outils de la thèse de Malod [57]. Pour le second, c'est un résultat de Bürgisser [15] qui nous permet de conclure.

Version unaire

Nous avons besoin de la notion de *circuits multiplicativement disjoints*, introduite par Malod [57, section 2.4], voir aussi Malod et Portier [59] (l'idée de restreindre le champ d'action des portes de multiplication apparaît déjà dans Toda [76] dans la notion encore plus restrictive des circuits *skew* et *weakly skew*).

2-E – Définition

Soient C un circuit arithmétique et α une porte dont les deux entrées sont les portes

> β et γ. La porte α est *disjointe* si les sous-circuits provenant de β et de γ sont disjoints.
>
> Le circuit C est *multiplicativement disjoint* si toutes ses portes de multiplication sont disjointes.

Ce type de circuit est intéressant en ceci qu'il permet d'utiliser la distributivité de la multiplication sur l'addition, comme on le verra ci-dessous. De plus, le résultat suivant de Malod et Portier [59, lemme 2] (déjà présent dans Malod [57]) montre de manière quantitative que tout circuit peut être rendu multiplicativement disjoint.

2-F – Lemme

> Soit C un circuit de taille t et de degré formel complet d. Il existe un circuit multiplicativement disjoint C' de taille $\leqslant dt$ qui calcule le même polynôme que C. De plus, on peut construire C' à partir de C en temps polynomial en $|C|$ et d.

2-G – Remarque Si d est donné en unaire (comme dans le problème $\mathrm{CMN_u}$), la construction de C' à partir de C se fait en temps polynomial.

Afin de montrer que $\mathrm{CMN_u}$ est dans $\mathsf{P}^{\#\mathsf{P}}$, on s'inspire de la section 2.4 de Malod [57] : le coefficient du monôme m dans un circuit multiplicativement disjoint est simplement le nombre de « développements » qui mènent à m. Un « développement » est un arbre qui représente les choix lors de la distributivité de $\times$ sur $+$. Plus précisément, si C est un circuit multiplicativement disjoint, un *développement* D est un sous-graphe de C qui satisfait les propriétés suivantes :

- toute porte $\times$ de D a ses deux entrées dans D ;
- toute porte $+$ de D a exactement une de ses entrées dans D ;
- D contient la sortie du circuit, et toute autre porte de D a au moins un fils dans D.

Le monôme calculé par un développement est simplement le produit des variables apparaissant en entrée. Si le développement contient la constante 0 en entrée alors on l'appellera *neutre*. Sinon, s'il contient la constante -1 on l'appellera *négatif* ; dans les autres cas, on dira qu'il est *positif*. Le résultat suivant de Malod [57] justifie l'introduction de la notion de développements.

2-H – Lemme

> Soit C un circuit multiplicativement disjoint et m un monôme $X_1^{\alpha_1} \cdots X_n^{\alpha_n}$. Soient $d^+(m)$ et $d^-(m)$ le nombre de développements positifs et négatifs calculant m, respectivement. Alors le coefficient de m dans le polynôme calculé par C est égal à $d^+(m) - d^-(m)$.

Cela nous permet de montrer le résultat suivant.

2-I – Lemme

> La fonction f qui, étant donnés un circuit multiplicativement disjoint C et un monôme m, calcule le coefficient de m, est dans GapP.

Preuve Vérifier qu'un sous-graphe est un développement calculant un monôme donné est dans P. Soit $A^+(m)$ (respectivement $A^-(m)$) l'ensemble des développements positifs (resp. négatifs) de C qui calculent m. Le coefficient de m est alors $|A^+(m)| -$

$|A^-(m)|$ et est calculable dans GapP car décider $A^+(m)$ et $A^-(m)$ se fait en temps polynomial. ∎

Ce lemme montre que décider si un coefficient est nul est dans $\mathsf{P}^{\#\mathsf{P}}$. Cela termine la preuve de la proposition 2-A.

Version générale

Puisque le cas unaire est réglé, attachons-nous au cas général. Nous montrons dans cette partie que le problème CMN est dans la hiérarchie de comptage CH.

La méthode précédente ne s'applique plus ici car on ne peut plus construire le circuit multiplicativement disjoint en temps polynomial, puisque le degré formel complet n'est plus donné en unaire.

L'idée est d'utiliser le polynôme « générique » G_n de Malod [57], dont il a calculé les coefficients et qui permet de simuler tout circuit arithmétique de taille inférieure à n. Ses coefficients sont des produits de coefficients binomiaux que l'on pourra alors évaluer dans la hiérarchie de comptage grâce à Bürgisser [15].

La construction du polynôme générique G_n s'effectue niveau par niveau, en partant des n variables $x_1, \ldots, x_n$. On définit d'abord

$$G_{-n}^{(n)} = 1, G_{-n+1}^{(n)} = x_1, \ldots, G_0^{(n)} = x_n.$$

Aux niveaux suivants $0 < l \leqslant n$, on calcule tous les produits et sommes possibles des niveaux inférieurs, en ajoutant de nouvelles variables $a_{l,i}$ et $b_{l,i}$:

$$G_l^{(n)} = \Big(\sum_{i=-n}^{l-1} a_{l,i} G_i^{(n)} \Big) \Big(\sum_{i=-n}^{l-1} b_{l,i} G_i^{(n)} \Big).$$

Enfin, on note $G_n(y_1, \ldots, y_p) = G_n^{(n)}$. Ce polynôme est « générique » en ce sens que tout circuit C de taille $\leqslant n$ est obtenu à partir de $G_n(y_1, \ldots, y_p)$ en spécialisant des variables : voici le résultat précis de Malod [57, Prop. 3].

2-J – Lemme

Soit $f(x_1, \ldots, x_k)$ un polynôme calculé par un circuit arithmétique C de taille $\leqslant n$.

1. Il existe $a_1, \ldots, a_p$ parmi $\{-1, 0, 1, 2, x_1, \ldots, x_k\}$ tels que
$$f(x_1, \ldots, x_k) = G_n(a_1, \ldots, a_p)\,;$$
2. les valeurs de $a_1, \ldots, a_p$ peuvent être calculées à partir de C en temps polynomial.

Si C est un circuit fixé et f le polynôme calculé par C, soit τ la substitution des variables $y_1, \ldots, y_p$ de G_n donnée par le lemme précédent, c'est-à-dire la fonction définie par $\tau(y_i) = a_i$. Nous écrirons $C = \tau(G_n)$.

Si m' est un monôme de G_n, l'image $\tau(m')$ de m' par la substitution τ est le produit d'un coefficient (éventuellement nul) et d'un monôme de f ; nous noterons $\alpha(m')$ le coefficient et $\tau'(m')$ le monôme, c'est-à-dire que l'on a $\tau(m') = \alpha(m')\tau'(m')$ où $\alpha(m')$ est un entier. On remarquera que τ' et α sont calculables en temps polynomial.

On a donc :

$$\mathrm{coef}_C(m) = \sum_{m' \mid \tau'(m')=m} \alpha(m')\mathrm{coef}_{G_n}(m').$$

Or les coefficients du polynôme G_n sont évalués à la section 5.2.3 de Malod [57], voir aussi l'article [58] : il s'agit de la proposition suivante.‡

2-K – Proposition

Le coefficient dans G_n du monôme

$$m = \prod_{\substack{1 \leqslant i \leqslant n \\ -n \leqslant j \leqslant n-1}} a_{i,j}^{\alpha_{i,j}} \prod_{\substack{1 \leqslant i \leqslant n \\ -n \leqslant j \leqslant n-1}} b_{i,j}^{\beta_{i,j}} \prod_{1 \leqslant i \leqslant n} x_i^{\gamma_i}$$

est nul s'il existe $i \geqslant 1$ tel que $\gamma_i \neq \sum_{j=1}^{n}(\alpha_{j,i} + \beta_{j,i})$; sinon il vaut

$$\prod_{i=1}^{n} \prod_{j=-n}^{i-1} \binom{\sum_{k=-n}^{j} \alpha_{i,k}}{\alpha_{i,j}} \binom{\sum_{k=-n}^{j} \beta_{i,k}}{\beta_{i,j}}.$$

Afin d'évaluer les coefficients de G_n, nous allons utiliser le résultat de Bürgisser [15] (théorème 3.7 et corollaire 3.8) montrant que les coefficients binomiaux sont calculables chiffre par chiffre dans la hiérarchie de comptage. Avant de donner le résultat précis, nous avons besoin de la définition d'une suite d'entier calculable chiffre par chiffre dans CH, issue de [15]. Nous en aurons de nouveau besoin à la partie 3.2, dans un contexte différent§, où nous approfondirons d'avantage cette notion ; nous reportons le lecteur à cette partie pour plus de détails.

2-L – Définition

Soit $p(n)$ un polynôme et $(a(n,k))_{n\in\mathbb{N},k\leqslant p(n)}$ une suite d'entiers vérifiant $a(n,k) \leqslant 2^{n^c}$ pour tout $k \leqslant p(n)$ et pour une certaine constante c. On dit que la suite $(a(n,k))$ *se calcule chiffre par chiffre dans* CH si le langage

$$\{(n,k,i,b) \mid \text{le } i\text{-ème bit de } a(n,k) \text{ est } b\}$$

est dans CH.

Voici enfin le résultat de Bürgisser [15] nous permettant de calculer les coefficients de G_n.

‡Par souci de modernité et de clarté, nous employons ici pour les coefficients binomiaux la notation ISO adoptée en France depuis 1994, aussi connue sous le nom de « notation anglo-saxonne » (que le lecteur encore habitué aux mathématiques de l'ancien millénaire, comme l'auteur de cette thèse, nous excuse).

§En effet, nous suivons ici les définitions de [15], mais à la partie 3.2 les entiers seront donnés en unaire et non plus en binaire.

2-M – THÉORÈME

1. *Le produit $a(N,k) = N(N-1)\cdots(N-k)$ se calcule chiffre par chiffre dans la hiérarchie de comptage.*
2. *Si $p(n)$ est un polynôme et $(a(N,k))$ une suite d'entiers qui se calcule chiffre par chiffre dans la hiérarchie de comptage, alors il en est de même de*

$$b(N) = \sum_{k=0}^{p(N)} a(N,k) \quad et \quad c(N) = \prod_{k=0}^{p(N)} a(N,k).$$

3. *Si $(s(N))$ et $(t(N))$ sont deux suites d'entiers qui se calculent chiffre par chiffre dans la hiérarchie de comptage, où $t(N) \neq 0$ pour tout N, alors le quotient $(\lfloor s(N)/t(N) \rfloor)$ se calcule chiffre par chiffre dans la hiérarchie de comptage.*

Ainsi, les coefficients binomiaux et donc les coefficients de G_n se calculent chiffre par chiffre dans la hiérarchie de comptage et il en est de même de la somme

$$\mathrm{coef}_C(m) = \sum_{m' \mid \tau'(m')=m} \alpha(m') \mathrm{coef}_{G_n}(m').$$

Il suffit maintenant d'un calcul coNP pour tester si tous les chiffres de ce coefficient sont nuls. Donc CMN $\in$ CH ce qui termine la preuve de la proposition 2-C.

2.4 Coefficient d'un monôme en caractéristique non nulle

Occupons-nous maintenant du cas de la caractéristique k non nulle. On travaille donc modulo k, c'est-à-dire que nos polynômes ont leurs coefficients dans $\mathbb{Z}/k\mathbb{Z}$. L'arithmétique modulaire va nous aider un peu car on parvient à caractériser complètement la complexité des problèmes CMN et CMN_u dans ce contexte. En effet, il nous est possible de calculer entièrement le coefficient d'un monôme car il n'y a plus de problème de taille des entiers. ¶

2-N – Proposition

Les problèmes CMN^k et $\mathrm{CMN}^k_\mathrm{u}$ sur un corps de caractéristique $k > 0$ sont $\mathsf{Mod}_k\mathsf{P}$-complets.

2-O – Remarque Le théorème de Toda [75] montre que $\mathsf{PH} \subseteq \mathsf{P}^{\oplus\mathsf{P}}$. Toda et Ogiwara [77] le généralisent à tout $k \geqslant 2$, c'est-à-dire

$$\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{Mod}_k\mathsf{P}}.$$

Ainsi, CMN^k et $\mathrm{CMN}^k_\mathrm{u}$ ne sont pas dans la hiérarchie polynomiale si elle est stricte.

Afin de montrer ces résultats de complétude, nous donnons une borne supérieure sur CMN^k et une borne inférieure sur $\mathrm{CMN}^k_\mathrm{u}$.

¶Rappel : pour spécifier qu'on travaille modulo k, on juchera un exposant k au-dessus du nom du problème.

2.4.1 Borne supérieure

L'algorithme que nous donnons ici pour CMN^k utilise le polynôme générique G_n décrit à la partie 2.3.2. Rappelons qu'il peut simuler n'importe quel circuit de taille $\leqslant n$ par substitution de certaines variables par les constantes -1, 0, 1 ou 2 (lemme 2-J). Soient C un circuit de taille $\leqslant n$ et τ la substitution correspondante, i.e. $C = \tau(G_n)$. Ainsi, les coefficients du polynôme calculé par C s'exprime en fonction de ceux de G_n : on a, pour un monôme m,

$$\mathrm{coef}_C(m) = \sum_{m' \mid \tau'(m')=m} \alpha(m')\mathrm{coef}_{G_n}(m').$$

Rappelons que l'on souhaite calculer le coefficient de m modulo k dans $\mathsf{Mod}_k\mathsf{P}$. Pour montrer que la somme précédente est dans $\mathsf{Mod}_k\mathsf{P}$, il suffit de montrer que les coefficients de G_n sont calculables en temps polynomial. En effet, on peut décider dans $\mathsf{Mod}_k\mathsf{P}$ si une somme exponentielle de termes calculables en temps polynomial est nulle modulo k. Montrons donc comment calculer les coefficients de G_n modulo k en temps polynomial. On utilise pour cela le vieux résultat suivant d'Édouard Lucas [55] sur les coefficients binomiaux.

2-P – Proposition

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 écrits en base k. On a

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

Les coefficients de G_n sont donc calculables en temps polynomial modulo k, ce qui prouve la première partie de la proposition 2-N.

2.4.2 Borne inférieure

On veut maintenant montrer que $\mathrm{CMN}^k_{\mathrm{u}}$ est dur pour $\mathsf{Mod}_k\mathsf{P}$. Pour cela, nous utilisons le problème suivant.

2-Q – Définition

On appelle $\mathrm{HamiltonCycle}^k$ le problème de décider si le nombre de cycles hamiltoniens dans un graphe est divisible par k.

L'intérêt de ce problème provient de sa complétude pour la classe $\mathsf{Mod}_k\mathsf{P}$.

2-R – Lemme

Le problème $\mathrm{HamiltonCycle}^k$ est $\mathsf{Mod}_k\mathsf{P}$-complet.

Preuve Le problème #HamiltonCycle consistant à compter le nombre de cycles hamiltoniens d'un graphe est #P-complet pour les réductions parsimonieuses (Valiant [81] ; on pourra aussi adapter la preuve de Papadimitriou [63, théorème 18.2] à des cycles), donc $\mathrm{HamiltonCycle}^k$, sa version modulo k, est $\mathsf{Mod}_k\mathsf{P}$-complète. ∎

Le nombre de cycles hamiltoniens dans un graphe de matrice d'adjacence $(x_{i,j})_{1\leqslant i,j\leqslant n}$ est donné par le polynôme

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

où la somme est prise sur tous les n-cycles $\sigma \in S_n$.

Ici encore, comme dans le cas du permanent, ce polynôme s'obtient comme coefficient d'un monôme d'un polynôme facilement calculable, comme le montre le lemme suivant de Malod [57, lemme 14].

2-S – Lemme

On définit les polynômes $T_{p,i,j}$ par $T_{1,i,j} = x_{i,j}y_iz_j$ et $T_{p+1,i,j} = \sum_{k=1}^{n} T_{p,i,k}T_{1,k,j}$. Alors

- le polynôme $P = T_{n,1,1}$ est calculable par un circuit arithmétique de taille $O(n^4)$ et de degré formel complet $O(n)$;
- ce circuit lui-même peut être construit en temps polynomial en n ;
- le coefficient du monôme $y_1z_1\cdots y_nz_n$ dans P est HC_n.

Nous déduisons de ce lemme que $\mathrm{CMN}_{\mathrm{u}}^k$ est plus difficile que $\mathrm{HamiltonCycle}^k$, ce qui conclut la preuve de la proposition 2-N.

2.5 Calcul du degré

On s'intéresse maintenant au calcul du degré d'un polynôme donné par un circuit, ou plus précisément aux langages DEG et $\mathrm{DEG}_{\mathrm{u}}$.

2.5.1 Version unaire

Commençons par le cas le plus simple, à savoir $\mathrm{DEG}_{\mathrm{u}}$. Pour ce problème, il existe un algorithme probabiliste simple qui fait appel à des techniques usuelles.

2-T – Proposition

Le problème $\mathrm{DEG}_{\mathrm{u}}$ est dans coRP.

Preuve Soit C un circuit, f le polynôme qu'il calcule et d un entier (donné en unaire, donc). On veut savoir si $\deg(f) \leqslant d$.

Pour cela, on commence par construire un circuit qui calcule toutes les composantes homogènes f_i de f de degré $i \leqslant d$. Cela se fait en temps polynomial (en d) en remplaçant chaque porte du circuit C par $d+1$ portes, la porte i calculant la composante de degré i. Il s'agit d'une construction classique, traitée par exemple dans Bürgisser [13, proposition 5.28] et Malod [57, lemme 2].

Afin de décider si f est de degré $\leqslant d$, il suffit maintenant de savoir si $f = \sum_{i=0}^{d} f_i$. Il s'agit donc de tester l'égalité de deux polynômes donnés par des circuits de taille polynomiale. Cela se fait par un algorithme probabiliste en testant l'égalité des deux circuits en des points aléatoires modulo des entiers aléatoires : c'est l'algorithme de

Schwartz [69]. Remarquons qu'on a besoin d'un grand nombre de points aléatoires dans le corps, ce qui peut poser un problème dans le cas d'un corps fini. Mais il suffit alors de travailler dans une extension. ∎

2.5.2 Version générale

Passons maintenant à l'étude du problème DEG. Sur un corps de caractéristique nulle, Allender, Bürgisser, Kjeldgaard-Pedersen et Miltersen [2] donnent une borne supérieure dans la hiérarchie de comptage pour ce problème, plus précisément dans $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$; nous n'améliorons pas ce résultat. En revanche, on peut faire un peu mieux sur un corps de caractéristique non nulle.

2-U – Proposition

- Les problèmes DEG (sur un corps de caractéristique nulle) et DEG^k (sur un corps de caractéristique $k > 0$) sont P-difficiles pour les réductions logspace.
- Le problème DEG^k sur un corps de caractéristique $k > 0$ est dans $\mathsf{coNP}^{\mathsf{Mod}_k\mathsf{P}}$.

Preuve Nous montrons les deux résultats successivement. Pour la borne inférieure, on réduit le problème PVC, qui est P-complet, à DEG^k. Le problème PVC, donné à la définition 1-H, consiste à savoir si un circuit booléen a pour valeur vraie ou non. Par le lemme 1-E, ce circuit booléen peut être simulé par un circuit arithmétique, dont la valeur est 1 si le circuit booléen est vrai, 0 sinon. Maintenant, on multiplie la sortie de ce circuit arithmétique par la variable x et on ajoute 1 : le polynôme calculé est de degré 0 si le circuit booléen est faux et de degré 1 sinon. La réduction se fait en espace logarithmique, donc DEG^k est P-difficile.

La borne supérieure repose sur la remarque suivante :

$$\deg(C) \leqslant d \text{ ssi (tout monôme de degré } > d \text{ a un coefficient nul).}$$

Puisque le nombre de monômes est simplement exponentiel, on peut tous les tester par un calcul coNP, ce qui donne

$$\mathrm{DEG}^k \in \mathsf{coNP}^{\mathrm{CMN}^k}.$$

Par la proposition 2-N, le problème CMN^k est dans $\mathsf{Mod}_k\mathsf{P}$, ce qui conclut. ∎

2-V – Remarque Allender nous indique que l'on peut également montrer la borne inférieure suivante : le problème DEG_u est SAC^1-difficile pour les réductions logspace. La classe SAC^1 définie dans [10] (nous parlons ici de sa version uniforme, aussi égale à l'ensemble LOGCFL des langages qui se réduisent en espace logarithmique à un langage algébrique) consiste en l'ensemble des langages reconnus par une famille uniforme de circuits booléens de profondeur logarithmique ayant des portes $\neg$ seulement aux entrées, des portes $\vee$ de degré entrant non-borné et des portes $\wedge$ de degré entrant borné. On peut alors faire la même preuve que précédemment mais en simulant $x \vee y$ par $x + y$, ce qui permet de restreindre le degré formel du circuit (remarquons que la simulation du circuit ne prend plus ses valeurs dans $\{0, 1\}$ mais prend une valeur 0 pour Faux et une valeur > 0 pour Vrai).

2.6 Résumé

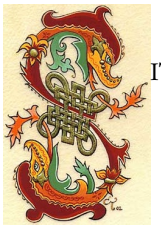
Voici un tableau récapitulatif des résultats de ce chapitre. La borne supérieure CH pour DEG est de Allender, Bürgisser, Kjeldgaard-Pedersen et Miltersen [2]. Les points d'interrogation signifient qu'aucun résultat non trivial n'a été obtenu. Une borne inférieure P veut dire que le problème est P-difficile pour les réductions logspace. La réduction $\leqslant_T^{sn}$ est définie au paragraphe 2.1.2, il s'agit d'une réduction non-déterministe à oracle. Les bornes inférieures $\mathsf{Mod}_k\mathsf{P}$ valent pour les réductions many-one.

	Caractéristique nulle		Caractéristique $k > 0$	
	Borne inférieure	Borne supérieure	Borne inf.	Borne sup.
CMN_u	$\mathsf{P}^{\#\mathsf{P}}$ (pour $\leqslant_T^{sn}$)	$\mathsf{P}^{\#\mathsf{P}}$	$\mathsf{Mod}_k\mathsf{P}$	
CMN		CH		
DEG_u	?	coRP	?	coRP
DEG	P	CH	P	$\mathsf{coNP}^{\mathsf{Mod}_k\mathsf{P}}$

Nous venons donc de voir que manipuler des polynômes donnés par des circuits n'est pas toujours chose aisée. Dans ce chapitre, les objets manipulés étaient directement les circuits arithmétiques. Dans le reste de ce document, nous allons plutôt nous intéresser aux relations entre classes de complexité. Pour commencer, dans le prochain chapitre nous étudions ce que des produits de taille exponentielle apportent au calcul de polynômes.

CHAPITRE
TROIS

Des produits de taille exponentielle

 ITUONS tout d'abord le contexte de ce chapitre ainsi que des suivants. À défaut de savoir résoudre les grandes questions de la théorie de Valiant, nous montrons les liens entre celle-ci et une autre théorie de complexité algébrique, le modèle de Blum, Shub et Smale (BSS) où il s'agit cette fois de problèmes de décision. Les liens que nous verrons s'appellent des théorèmes de transfert et sont de la forme « séparer deux classes dans le modèle BSS implique la séparation de deux classes dans le modèle de Valiant ». C'est assez cohérent avec notre intuition, d'abord car le modèle de Valiant est plus simple et ensuite car nous pensons que ces deux séparations sont vraies pour les classes que nous étudierons. Cela suggère d'étudier en premier lieu les questions de complexité dans le modèle de Valiant car elles semblent moins difficiles.

Dans ce chapitre, sur un corps de caractéristique nulle, nous nous intéressons à la question « $\mathsf{VP} = \mathsf{VNP}\,?$ » d'une part, et à « $\mathsf{NP}_{(K,+)} \subseteq \mathsf{P}_K\,?$ » (c'est-à-dire une version faible de $\mathsf{P}_K = \mathsf{NP}_K$) d'autre part. Nous montrons que la première égalité implique la seconde.

Travaux précédents. Fournier et Koiran [27] et [28] ont déjà montré des résultats de transfert vers le modèle BSS mais en partant de classes de complexité booléenne. En particulier, ils ont montré que $\mathsf{P}^0_{(\mathbb{R},+,<)} = \mathsf{NP}^0_{(\mathbb{R},+,<)}$ si $\mathsf{P} = \mathsf{NP}$. À première vue, ce résultat semble plus fort que celui de ce chapitre car il prend en compte l'ordre sur $\mathbb{R}$ et part d'une hypothèse plus faible (comme nous le verrons au théorème 3-E). Cependant, il semble que l'ordre soit au contraire une aide pour ce genre de résultat car il permet d'effectuer une recherche dichotomique.

Ici, nous montrons un résultat du même genre sur un corps non-ordonné mais en partant d'une hypothèse de théorie de Valiant. Cette hypothèse permet d'effectuer la recherche dichotomique qui devenait difficile sans l'ordre. La conclusion du résultat est asymétrique puisqu'on autorise la multiplication pour P mais pas pour NP. Cette asymétrie est plus naturelle qu'il n'y paraît, car Meer [60] a séparé P et NP sur la structure non-ordonnée $(\mathbb{R},+)$: il semblait donc peu probable de ne pas avoir à ajouter la multiplication à P…

Produits de taille exponentielle. Pour montrer notre théorème de transfert, nous passons par l'intermédiaire de produits de taille exponentielle. Jusqu'à présent, les sommes exponentielles dans le modèle de Valiant ont été largement étudiées, mais personne ne semble avoir étudié les produits exponentiels : cette injustice est enfin sur le point d'être réparée. Un exemple de produit de taille exponentielle, détaillé dans l'exemple 3-B, est la famille de polynômes (P_n) définie par

$$P_n = \prod_{i=0}^{2^n} (X - i).$$

On conjecture que cette famille est difficile à calculer, *cf* Heintz et Morgenstern [32] ou Lipton [52]. De plus, la τ-conjecture de Shub et Smale [70][‖] suggère que le nombre de racines entières d'un polynôme à une variable est polynomial en la taille du plus petit circuit le calculant. Tout cela laisse penser que les gros produits ne sont pas calculables par des circuits de taille polynomiale. Le théorème 3-E ci-dessous est peut-être encore plus probant : si c'était le cas, alors $\mathsf{P/poly} = \mathsf{NP/poly}$ (ce qui implique que la hiérarchie polynomiale s'effondre au second niveau, *cf* Karp et Lipton [39]).

Ainsi, les produits de taille exponentielle semblent être difficiles à calculer. Mais ils peuvent nous apprendre des choses sur les problèmes de décision. Prenons l'exemple du problème Twenty Questions sur $\mathbb{C}$ de Shub et Smale [70] : sur l'entrée constituée d'un nombre complexe x et d'un entier d donné en binaire, il s'agit de décider si $x \in \{0, 1, \ldots, d-1\}$. Il est facile de voir que Twenty Questions est dans $\mathsf{NP}_{\mathbb{C}}$: il suffit de deviner l'écriture binaire de x. Bien que ce problème ne semble pas être $\mathsf{NP}_{\mathbb{C}}$-complet, Shub et Smale [70] ont donné des indices suggérant qu'il n'est pas dans $\mathsf{P}_{\mathbb{C}}$. Ainsi, c'est un candidat très simple pour montrer que $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$.

Le rapport avec nos produits de taille exponentielle est le suivant : si l'on sait évaluer en temps polynomial le polynôme P_n de l'exemple précédent, alors pour savoir si $x \in \{0, 1, \ldots, 2^n\}$ il suffit de tester si $P_n(x) = 0$. Ainsi, Twenty Questions est dans $\mathsf{P}_{\mathbb{C}}$ si l'on sait calculer en temps polynomial les produits de taille exponentielle. En d'autres termes, si l'on parvient à séparer $\mathsf{NP}_{\mathbb{C}}$ de $\mathsf{P}_{\mathbb{C}}$ grâce à Twenty Questions (c'est-à-dire à montrer que Twenty Questions n'est pas dans $\mathsf{P}_{\mathbb{C}}$) alors on aura montré que les produits de taille exponentielle ne sont pas facilement calculables.

Nous venons de traiter le cas de Twenty Questions grâce à de gros produits car il s'y prêtait particulièrement bien ; peut-on en faire autant avec d'autres problèmes ? Le résultat principal de ce chapitre montre que l'on peut généraliser l'argument à la classe $\mathsf{NP}_{(\mathbb{C},+)}$ des problèmes sur $\mathbb{C}$ décidables en temps polynomial non-déterministe pour lesquels la multiplication est interdite. Cette classe contient entre autres Twenty Questions et Subset Sum. Ainsi, si l'on sépare $\mathsf{P}_{\mathbb{C}}$ de $\mathsf{NP}_{\mathbb{C}}$ en utilisant un des problèmes de cette classe, on aura montré que les produits exponentiels ne sont pas calculables en temps polynomial. Cette dernière question est donc moins difficile, ce qui suggère de s'y intéresser en premier.

On voit donc que l'étude du calcul de polynômes permet d'obtenir des résultats sur les problèmes de décision. Ici nous nous intéressons aux produits de taille exponentielle, mais on peut relier tout cela avec les classes de Valiant plus connues : dans la partie 3.2,

[‖] Dans l'article [70], la τ-conjecture n'apparaît pas explicitement mais toutes les idées sont présentes. L'énoncé explicite peut être trouvé dans le livre de Blum, Cucker, Shub et Smale [8, chapitre 7] ou dans Smale [71, problème 4].

on montrera que si VP et VNP coïncident, alors les gros produits sont calculables par des circuits de taille polynomiale. Il est donc plus facile de séparer VP et VNP que de montrer que les gros produits sont difficiles à calculer. Finalement, la question la plus simple est l'hypothèse de Valiant !

Plan du chapitre. Nous commençons à la partie 3.1 par définir le cadre de l'étude, à savoir la classe VΠP des produits de taille exponentielle. Dans cette partie également, le théorème 3-E montre que si VΠP a des circuits de taille polynomiale, alors P/poly et NP/poly coïncident. La partie suivante prouve que $\mathsf{VP} = \mathsf{VNP}$ implique $\mathsf{VP}_{\mathsf{nb}} = \mathsf{V\Pi P}$. Cela repose sur des polynômes d'interpolation, que l'on peut calculer grâce aux résultats de Bürgisser [15]. Enfin, la partie 3.3 est consacrée au résultat principal, à savoir le résultat de transfert vers le modèle BSS. Le principal outil utilisé sera la localisation d'un point dans un arrangement d'hyperplans.

Les résultats de ce chapitre viennent de l'article [44] pour le théorème de transfert et de [46] pour la partie 3.2 sur les liens entre l'hypothèse de Valiant et les gros produits.

3.1 Des produits exponentiels

Dans cette partie, nous définissons la classe VΠP des familles de polynômes qui sont des produits de taille exponentielle et en donnons quelques propriétés. Puis nous voyons une conséquence en complexité booléenne de l'hypothèse que ces gros produits sont facilement calculables.

3.1.1 Définition

La classe VΠP est très inspirée de la définition de VNP, mais la somme est remplacée par un produit. Nous nous plaçons dans un contexte de calcul uniforme et sans constante ; on adaptera sans difficulté la définition aux calculs non-uniformes et avec constantes.

3-A – Définition

La classe Uniform $\mathsf{V\Pi P}^0$ est l'ensemble des familles $(g_n(\bar{x}))$ de polynômes telles qu'il existe une famille $(f_n(\bar{x},\bar{y})) \in \mathsf{Uniform\ VP}^0_{\mathsf{nb}}$ satisfaisant

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

On remarquera que l'hypothèse sur (f_n) impose une borne polynomiale sur la taille de $\bar{x}$ et $\bar{y}$.

Comme promis dans l'introduction, voyons un exemple de famille $\mathsf{V\Pi P}^0$.

3-B – Exemple La famille (P_n) définie par $P_n(X) = \prod_{i=0}^{2^n-1}(X-i)$ est dans Uniform $\mathsf{V\Pi P}^0$.

Cela vient de l'égalité

$$P_n(X) = \prod_{\bar{\epsilon}\in\{0,1\}^{n-1}} f_n(X,\bar{\epsilon}),$$

où la famille (f_n) définie par $f_n(X,\bar{y}) = X - \sum_{i=1}^{n} y_i 2^{i-1}$ est dans Uniform $\mathsf{VP}^0_{\mathsf{nb}}$.

3-C – Remarque Tout comme pour les classes de Valiant usuelles, le corps sous-jacent est implicite. On peut considérer qu'on a fixé une fois pour toute un corps K de caractéristique nulle. Remarquons que le choix du corps n'affecte pas la réponse à la question « $\mathsf{VP}^0_{\mathsf{nb}} = \mathsf{V\Pi P}^0$? », car elle est identique dans tous les corps de caractéristique nulle.

3.1.2 Premières propriétés

Pour les développements techniques, nous allons avoir besoin de faire des produits sur des ensembles plus compliqués que $\{0,1\}^{p(n)}$. C'est l'objet du lemme suivant.

3-D – Lemme

Soient $(f_n(\bar{x},\bar{y}))$ une famille Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ et $s(n)$ une fonction polynomialement bornée telle que $|\bar{y}| \leqslant s(n)$. Soit A un langage dans P. Alors il existe une famille $(g_n(\bar{\ell},\bar{x})) \in$ Uniform $\mathsf{V\Pi P}^0$, où $|\bar{\ell}| = s(n) - |\bar{y}|$, telle que pour tout uple booléen $\bar{\ell}$, on ait

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

Preuve Puisque $A \in \mathsf{P}$, il existe une famille uniforme de circuits booléens (C_n) de taille polynomiale qui décide A. Par le lemme 1-E, on peut simuler ces circuits par des circuits arithmétiques de taille polynomiale : on obtient une famille de polynômes $(c_n(\bar{y},\bar{z}))$ dans Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ telle que sur toute entrée booléenne $(\bar{\ell},\bar{\epsilon})$ de taille n, on ait :

$$c_n(\bar{\ell},\bar{\epsilon}) = \begin{cases} 1 \text{ si } (\bar{\ell},\bar{\epsilon}) \in A \\ 0 \text{ sinon.} \end{cases}$$

La famille $(h_n(\bar{x},\bar{y},\bar{z}))$ définie par

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

est donc dans Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ et satisfait

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

3.1.3 Liens avec la complexité booléenne

Le fait que les gros produits soient calculables par des circuits de taille polynomiale, c'est-à-dire l'égalité $\mathsf{VP}^0_{\mathsf{nb}} = \mathsf{V\Pi P}^0$, semble bien peu probable comme le montre le théorème suivant. L'idée est de remplacer les quantificateurs de NP par un gros produit. Attention, le contexte ici est non-uniforme car le résultat s'y prête mieux.

3-E – THÉORÈME

Si $\mathsf{VP}^0_{\mathsf{nb}} = \mathsf{V\Pi P}^0$ *alors* $\mathsf{P/poly} = \mathsf{NP/poly}$.

Preuve Soit $A \in \mathsf{NP/poly}$: il existe un langage $B \in \mathsf{P/poly}$ et un polynôme $p(n)$ tels que

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

Puisque $B \in \mathsf{P/poly}$, il est décidé par une famille de circuits booléens de taille polynomiale. Ces circuits peuvent être simulés par des circuits arithmétiques, grâce au lemme 1-E. On obtient donc une famille de polynômes $(f_n(\bar{x},\bar{y}))$ qui, lorsqu'on l'évalue sur une entrée booléenne $(\bar{x},\bar{y})$, vaut 0 si $(\bar{x},\bar{y}) \in B$ et 1 sinon. Cette famille est dans $\mathsf{VP}^0_{\mathsf{nb}}$ car les circuits ont une taille polynomiale.
Maintenant, la famille de produits

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

est dans $\mathsf{V\Pi P}^0$. Sur toute entrée booléenne $\bar{x}$, on a $g_n(\bar{x}) \in \{0,1\}$, et $g_n(\bar{x}) = 0$ si et seulement si $\exists \bar{y} \in \{0,1\}^{p(|\bar{x}|)}.(f_n(\bar{x},\bar{y}) = 0)$. En d'autres termes,

$$g_n(\bar{x}) = 0 \Leftrightarrow \bar{x} \in A. \tag{3.1}$$

Sous l'hypothèse $\mathsf{V\Pi P}^0 = \mathsf{VP}^0_{\mathsf{nb}}$, la famille (g_n) est dans $\mathsf{VP}^0_{\mathsf{nb}}$. Elle est donc calculée par des circuits arithmétiques de taille polynomiale. Ainsi, décider si $\bar{x} \in A$ en temps polynomial non-uniforme revient à tester en temps polynomial non-uniforme si la valeur d'un circuit est nulle. C'est un problème classique : on peut le faire dans coRP en calculant modulo des nombres premiers aléatoires (voir par exemple Schwartz [69]). L'inclusion $\mathsf{coRP} \subset \mathsf{P/poly}$ (Adleman [1]) conclut la preuve. ∎

L'équivalence (3.1) montre que pour décider un problème NP, il suffit d'évaluer une famille Uniform $\mathsf{V\Pi P}^0$. On utilisera cette propriété dans la partie 3.3.6.

3.2 Liens avec l'hypothèse de Valiant

Nous montrons ici que la question « $\mathsf{VP}^0_{\mathsf{nb}} \neq \mathsf{V\Pi P}^0$? » et l'hypothèse de Valiant « $\mathsf{VP}^0 \neq \mathsf{VNP}^0$? »sont liées. En effet, nous avons le résultat suivant.

3-F – THÉORÈME

Si Uniform VP^0 = Uniform VNP^0 *alors* Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ = Uniform $\mathsf{V\Pi P}^0$.

Le reste de la section est consacrée à la preuve de ce résultat. Pour cela, la méthode que nous allons utiliser repose sur l'interpolation de Lagrange, que l'on pourra rendre effective grâce aux résultats de Bürgisser [15] sur la hiérarchie de comptage.

Dans un premier temps, nous définissons une notion de suite d'entiers calculable dans la hiérarchie de comptage et de polynôme évaluable aux points entiers. Puis nous appliquerons cela à des polynômes d'interpolation.

3.2.1 Suites d'entiers

Afin d'éviter de manipuler le signe d'entiers séparément, nous supposerons que le codage booléen de tels entiers contient leur signe comme premier bit, par exemple, suivi du code de la valeur absolue de l'entier.

Voici la notion d'entiers calculables dans la hiérarchie de comptage que nous allons utiliser. Attention, ce n'est pas la même notion que dans Bürgisser [15] ou qu'au chapitre 2 où les entiers sont codés en binaire.

3-G – Définition

Une *suite d'entiers de taille exponentielle* est une suite d'entiers $(a(n,k))$ telle qu'il existe un polynôme $p(n)$ qui satisfait :

1. l'entier $a(n,k)$ est défini pour $n,k \in \mathbb{N}$ et $0 \leqslant k < 2^{p(n)}$;
2. pour tout $n > 1$ et tout $k < 2^{p(n)}$, la taille de l'encodage binaire de $a(n,k)$ est $\leqslant 2^{p(n)}$.

On définit alors à partir de $a(n,k)$ le langage suivant :

$$\mathrm{Bit}(a) = \{(1^n, k, j, b) | \text{ le } j\text{-ème bit de } a(n,k) \text{ est } b\}.$$

Ainsi, c'est une définition analogue à Bürgisser [15] mais n est ici donné en unaire.

3-H – Définition

Une suite d'entiers $a(n,k)$ de taille exponentielle est *définissable dans* CH si le langage $\mathrm{Bit}(a)$ est dans CH.

3-I – Remarque On rencontrera aussi des suites avec plus de deux paramètres (n,k), par exemple $a(n,\alpha^{(1)},\ldots,\alpha^{(n)})$ pour des entiers $\alpha^{(i)}$. Afin de les voir comme des suites à deux paramètres, $(\alpha^{(1)},\ldots,\alpha^{(n)})$ sera considéré comme l'encodage d'un unique entier. Par abus de notation, le paramètre n pourra aussi être donné en indice, comme dans $f_n(k)$ par exemple, alors que l'on devrait écrire $f(n,k)$.

Nous pouvons donner une définition similaire pour des familles de polynômes.

3-J – Définition

Soit $(f_n(x_1,\ldots,x_{u(n)}))$ une famille de polynômes à coefficients entiers. On dit que (f_n) peut être *évaluée dans* CH *aux points entiers* si les conditions suivantes sont vérifiées :

1. le nombre $u(n)$ de variables est polynomialement borné ;
2. le degré de f_n ainsi que la taille binaire de ses coefficients sont bornés par $2^{p(n)}$ pour un certain polynôme $p(n)$;
3. le langage $\{(1^n, i_1,\ldots,i_{u(n)}, j, b) |$ le j-ème bit de $f_n(i_1,\ldots,i_{u(n)})$ est $b\}$ est dans CH.

Au vu de ces deux définitions, le lemme suivant est alors clair.

3-K – Lemme

La famille $(f_n(x_1,\ldots,x_{u(n)}))$ peut être évaluée dans CH aux points entiers si et seulement si la suite d'entiers $a(n,i_1,\ldots,i_{u(n)}) = f_n(i_1,\ldots,i_{u(n)})$ est définissable dans CH.

Par ailleurs, nous aurons besoin du théorème suivant d'Allender, Bürgisser, Kjeldgaard-Pederson et Miltersen [2, théorème 4.1], ou plus précisément de son corollaire 3-M.

3-L – THÉORÈME

Soit BitSLP *le problème suivant : étant donnés un entier $i \in \mathbb{N}$ en binaire ainsi qu'un circuit arithmétique sans constante qui calcule un entier N, décider si le i-ème bit de la représentation binaire de N est 1. Le problème* BitSLP *est dans* CH.

3-M – Corollaire

Soit $(f_n) \in \mathsf{VP}^0_{\mathsf{nb}}$. Alors (f_n) peut être évalué dans CH aux points entiers.

Les résultats de cette partie reposerons sur le lien suivant entre les classes de Valiant et la hiérarchie de comptage, version uniforme de Bürgisser [15, lemmes 2.5 et 2.12]

3-N – Lemme

Si Uniform $\mathsf{VP}^0 =$ Uniform VNP^0 alors $\mathsf{CH} = \mathsf{P}$.

En particulier, ce lemme a été utilisé par Bürgisser [15, théorème 3.7] pour montrer que les sommes et produits de taille exponentielle sont calculables dans la hiérarchie de comptage. Nous en donnons ici une version où les entiers sont donnés en unaire : il s'agit d'une simple « mise à l'échelle » de [15, théorème 3.7] donné au théorème 2-M (il suffit de définir $a'(2^{p(n)}, k) = a(n,k)$ et d'appliquer le résultat de Bürgisser).

3-O – THÉORÈME

1. *Soient $p(n)$ un polynôme et $a = (a(n,k))_{n\in\mathbb{N},k\leqslant 2^{p(n)}}$ une suite définissable dans* CH. *Considérons*

$$b(n) = \sum_{k=0}^{2^{p(n)}} a(n,k) \quad et \quad d(n) = \prod_{k=0}^{2^{p(n)}} a(n,k).$$

   Alors $(b(n))_{n\in\mathbb{N}}$ et $(d(n))_{n\in\mathbb{N}}$ sont aussi définissables dans CH.

2. *Supposons que $(s(n))_{n\in\mathbb{N}}$ et $(t(n))_{n\in\mathbb{N}}$ soient définissables dans* CH *et que $t(n) > 0$. Alors la suite des quotients $(\lfloor s(n)/t(n) \rfloor)_{n\in\mathbb{N}}$ est aussi définissable dans* CH.

Nous pouvons maintenant commencer la preuve du théorème 3-F.

3.2.2 Coefficients

Le lemme suivant est le critère de Valiant [79], voir par exemple Bürgisser [13, proposition 2.20] et Koiran [43, théorème 2.3].

3-P – Lemme

Soit $a : (1^n, i) \mapsto a(1^n, i)$ une fonction de GapP, où n est donné en unaire et i en binaire. Soit $p(n)$ un polynôme et définissons la suite de polynômes suivante :

$$f_n(x_1, \ldots, x_{p(n)}) = \sum_{i=0}^{2^{p(n)}-1} a(1^n, i) x_1^{i_1} \cdots x_{p(n)}^{i_{p(n)}},$$

où i_j est le j-ème bit dans l'écriture binaire de i.
Alors $(f_n) \in \text{Uniform VNP}^0$.

Passons maintenant à une généralisation de [15, théorème 4.1(2)] aux polynômes à plusieurs variables, mise à l'échelle encore une fois.

3-Q – Lemme

Soit

$$f_n(x_1,\ldots,x_n) = \sum_{\alpha^{(1)},\ldots,\alpha^{(n)}} a(n,\alpha^{(1)},\ldots,\alpha^{(n)}) x_1^{\alpha^{(1)}} \cdots x_n^{\alpha^{(n)}},$$

où les entiers $\alpha^{(i)}$ varient de 0 à $2^n - 1$ et $a(n,\alpha^{(1)},\ldots,\alpha^{(n)})$ est une suite d'entiers définissable dans CH et de valeur absolue $< 2^{2^n}$.
Si Uniform $\text{VP}^0 = \text{Uniform VNP}^0$ alors $(f_n) \in \text{Uniform VP}^0_{\text{nb}}$.

Preuve Considérons l'écriture binaire de a : $a(n,\alpha^{(1)},\ldots,\alpha^{(n)}) = \sum_{i=0}^{2^n} a_i(n,\bar{\alpha}) 2^i$. Soit h_n le polynôme suivant :

$$h_n(x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}, z_1, \ldots, z_n) =$$

$$\sum_{i=0}^{2^n} \sum_{\bar{\alpha}} a_i(n,\bar{\alpha}) z_1^{i_1} \cdots z_n^{i_n} x_{1,1}^{\alpha_1^{(1)}} x_{1,2}^{\alpha_2^{(1)}} \cdots x_{1,n}^{\alpha_n^{(1)}} x_{2,1}^{\alpha_1^{(2)}} \cdots x_{n,n}^{\alpha_n^{(n)}}.$$

On a alors :

$$h_n(x_1^{2^0}, x_1^{2^1}, \ldots, x_1^{2^n}, x_2^{2^0}, \ldots, x_n^{2^n}, 2^{2^0}, 2^{2^1}, \ldots, 2^{2^n}) = f_n(x_1,\ldots,x_n).$$

Puisque $\text{VP}^0 = \text{VNP}^0$, par le lemme 3-N la hiérarchie de comptage s'effondre, donc calculer le i-ème bit $a_i(n,\bar{\alpha})$ de $a(n,\bar{\alpha})$ sur l'entrée $(1^n,\bar{\alpha},i)$ est dans GapP (et même dans P). Par le lemme 3-P, $(h_n) \in \text{VNP}^0$. Enfin, par l'hypothèse $\text{VP}^0 = \text{VNP}^0$, $(h_n) \in \text{VP}^0$ et on obtient $(f_n) \in \text{VNP}^0_{\text{nb}}$ en calculant les grandes puissances par élévations au carré successives. ∎

3.2.3 Interpolation

Voyons maintenant deux lemmes sur l'interpolation de Lagrange pour des polynômes à plusieurs variables.

3-R – Lemme

Soit $p(x_1,\ldots,x_n)$ un polynôme de degré $\leqslant d$. Alors

$$p(x_1,\ldots,x_n) = \sum_{0 \leqslant i_1,\ldots,i_n \leqslant d} p(i_1,\ldots,i_n) \prod_{k=1}^{n} \Big(\prod_{j_k \neq i_k} \frac{x_k - j_k}{i_k - j_k} \Big),$$

où les entiers j_k varient de 0 à d.

Preuve On procède par récurrence sur le nombre n de variables. Pour $n = 1$, c'est l'interpolation de Lagrange habituelle : on a

$$p(x) = \sum_{i=0}^{d} p(i) \prod_{j \neq i} \frac{x - j}{i - j}$$

car les deux polynômes sont de degré $\leqslant d$ et coïncident sur au moins $d + 1$ points distincts.

Pour $n + 1$, l'hypothèse de récurrence au rang $n = 1$ fournit

$$p(x_1, \ldots, x_{n+1}) = \sum_{i_{n+1}=0}^{d} p(x_1, \ldots, x_n, i_{n+1}) \prod_{j_{n+1} \neq i_{n+1}} \frac{x_{n+1} - j_{n+1}}{i_{n+1} - j_{n+1}}.$$

Par récurrence au rang n cette fois, c'est équivalent à

$$\sum_{i_{n+1}=0}^{d} \left(\sum_{0 \leqslant i_1, \ldots, i_n \leqslant d} p(i_1, \ldots, i_n) \prod_{k=1}^{n} \prod_{j_k \neq i_k} \frac{x_k - j_k}{i_k - j_k} \right) \prod_{j_{n+1} \neq i_{n+1}} \frac{x_{n+1} - j_{n+1}}{i_{n+1} - j_{n+1}},$$

ce qui est le résultat désiré. ∎

3-S – Lemme

Soit $a(n) = \prod_{i=0}^{2^n - 1} \prod_{j \neq i} (i - j)$, où j varie de 0 à $2^n - 1$. Soit $p_{i_1, \ldots, i_n}(\bar{x})$ la famille de polynômes suivante :

$$p_{i_1, \ldots, i_n}(x_1, \ldots, x_n) = \prod_{k=1}^{n} \left(a(n) \prod_{j_k \neq i_k} \frac{x_k - j_k}{i_k - j_k} \right),$$

où les entiers j_k varient de 0 à $2^n - 1$ et les entiers i_k sont donnés en binaire et varient de 0 à $2^n - 1$. Alors les coefficients de $p_{i_1, \ldots, i_n}$ sont des entiers définissables dans la hiérarchie de comptage, tout comme $a(n)$.

Preuve Dans un premier temps, remarquons que le coefficient du monôme $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ dans p_n est égal au produit des coefficients des monômes $x_k^{\alpha_k}$ dans les polynômes à une variable $a(n) \prod_{j_k \neq i_k} \frac{x_k - j_k}{i_k - j_k}$. Ainsi, nous avons juste à vérifier que ces différents coefficients de polynômes à une variable sont eux-mêmes définissables dans la hiérarchie de comptage. Concentrons-nous d'abord sur le polynôme à une variable $\prod_{j_k \neq i_k} (x_k - j_k)$, c'est-à-dire que nous oublions momentanément le terme multiplicatif $b(n, i_k) = a(n) / \prod_{j_k \neq i_k} (i_k - j_k)$.

Nous utilisons le même argument que Bürgisser [15, corollaire 3.9]. Remarquons que les coefficients de ce polynôme sont bornés en valeur absolue par $2^{2^{n^2}}$. Ainsi, dans le polynôme à une variable $\prod_{j_k \neq i_k} (x_k - j_k)$, on peut remplacer la variable x_k par $2^{2^{n^2}}$ et il n'y aura pas de recouvrement des coefficients des différentes puissances de x_k. On pourra donc retrouver les coefficients du monôme à partir de la valeur

de ce produit. Par le premier point du théorème 3-O, on peut évaluer dans CH le polynôme au point $2^{2^{n^2}}$, car il s'agit d'un produit de taille exponentielle. Ainsi, les coefficients sont définissables dans la hiérarchie de comptage.

Il suffit maintenant de remarquer que le premier point du théorème 3-O implique que $a(n)$ ainsi que $b(n,i_k)=a(n)/\prod_{j_k\neq i_k}(i_k-j_k)$ sont également définissables dans la hiérarchie de comptage. ∎

On notera au passage que la suite $a(n)$ du lemme 3-S est introduite seulement afin d'obtenir des coefficients entiers. On devra ensuite diviser par $a(n)$ dans les preuves suivantes.

3.2.4 Résultat principal

On peut maintenant énoncer le théorème principal de cette partie.

3-T – THÉORÈME

Soit $(f_n(x_1,\dots,x_{u(n)}))$ *une famille de polynômes à plusieurs variables. Supposons que* (f_n) *puisse être évaluée dans* CH *aux points entiers. Alors, sous l'hypothèse* Uniform $\mathsf{VP}^0=$ Uniform VNP^0*, on a* $(f_n)\in$ Uniform $\mathsf{VP}^0_{\mathsf{nb}}$.

Preuve L'objectif est de calculer le polynôme d'interpolation du lemme 3-R :

$$f_n(x_1,\dots,x_{u(n)})=\sum_{0\leqslant i_1,\dots,i_{u(n)}\leqslant d} f_n(i_1,\dots,i_{u(n)})\prod_{k=1}^{u(n)}\Big(\prod_{j_k\neq i_k}\frac{x_k-j_k}{i_k-j_k}\Big).$$

Par hypothèse, (f_n) peut être évalué dans CH aux points entiers. Cela implique par le lemme 3-R que $f_n(i_1,\dots,i_{u(n)})$ est définissable dans CH. On souhaite appliquer le lemme 3-S afin de calculer les coefficients de f_n dans CH. Il suffit de remarquer que le polynôme $p_{i_1,\dots,i_n}$ et la suite $a(n)$ du lemme 3-S satisfont

$$f_n(i_1,\dots,i_{u(n)})\prod_{k=1}^{u(n)}\prod_{j_k\neq i_k}\frac{x_k-j_k}{i_k-j_k}=a(n)^{-n}f_n(i_1,\dots,i_{u(n)})p_{i_1,\dots,i_{u(n)}}(\bar{x}).$$

Par le lemme 3-S et par le second point du théorème 3-O pour la division par $a(n)^n$, les coefficients de f_n sont définissables dans CH. Ainsi, par le lemme 3-Q, $(f_n)\in$ Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ sous l'hypothèse Uniform $\mathsf{VP}^0=$ Uniform VNP^0. ∎

On en déduit quelques corollaires.

3-U – Corollaire

Soit $(f_n(\bar{x},\bar{\epsilon}))\in$ Uniform $\mathsf{VP}^0_{\mathsf{nb}}$. Soient

$$g_n(\bar{x})=\sum_{\bar{\epsilon}} f_n(\bar{x},\bar{\epsilon}) \text{ et } h_n(\bar{x})=\prod_{\bar{\epsilon}} f_n(\bar{x},\bar{\epsilon}).$$

Si Uniform $\mathsf{VNP}^0=$ Uniform VP^0 alors (g_n) et (h_n) sont dans Uniform $\mathsf{VP}^0_{\mathsf{nb}}$.

Preuve Par le corollaire 3-M, (f_n) peut être évalué dans CH aux points entiers. En utilisant le lemme 3-R avant et après le premier point du théorème 3-O, on voit que (g_n) et (h_n) peuvent aussi être évalués dans CH aux points entiers. Le résultat découle alors du théorème 3-T. ∎

On obtient alors le corollaire suivant, dont surtout le second point nous concerne dans ce chapitre.

3-V – Corollaire

1. Uniform $\mathsf{VP}^0 = \text{Uniform } \mathsf{VNP}^0 \Longrightarrow \text{Uniform } \mathsf{VP}^0_{\mathsf{nb}} = \text{Uniform } \mathsf{VNP}^0_{\mathsf{nb}}$;
2. Uniform $\mathsf{VP}^0 = \text{Uniform } \mathsf{VNP}^0 \Longrightarrow \text{Uniform } \mathsf{VP}^0_{\mathsf{nb}} = \text{Uniform } \mathsf{V\Pi P}^0$.

Ainsi, montrer que les produits de taille exponentielle ne sont pas calculables par des circuits de taille polynomiale implique de montrer l'hypothèse de Valiant, ce qui ne semble pas simple.

Nous avons donc le lien entre le calcul de produits exponentiels et celui de sommes exponentielles. Au passage, nous avons montré que séparer les versions non-bornées de VP et VNP était au moins aussi difficile que de séparer les versions bornées, un résultat qui était déjà connu sur un corps de caractéristique non nulle : voir Malod [57].

3.3 Le théorème de transfert

Nous allons maintenant tisser des liens avec les problèmes de décision dans le cadre du modèle BSS. Le théorème de transfert que nous allons montrer est le suivant.

3-W – THÉORÈME

Si Uniform $\mathsf{VP}^0_{\mathsf{nb}} = \text{Uniform } \mathsf{V\Pi P}^0$ *alors* $\mathsf{NP}_{(K,+)} \subseteq \mathsf{P}_K$.

Une façon d'interpréter ce résultat, à rapprocher du théorème 3-E, est que les gros produits ont la puissance d'une quantification existentielle. Ce résultat a ceci de surprenant que supposer que les gros produits sont faciles à calculer implique en fait que la multiplication donne à P_K la puissance de $\mathsf{NP}_{(K,+)}$.

Avant d'aborder la preuve de ce résultat, il se pourrait que certaines notations nécessitent quelques rappels sur les classes BSS…

3.3.1 Rappels

Dans le modèle BSS on s'intéresse à des langages sur le corps K. Nous rappelons que la classe P_K est alors définie par l'ensemble des langages reconnus par une famille uniforme de circuits algébriques de taille polynomiale, et NP_K est la version existentielle de P_K. La structure sous-jacente est $(K, +, -, \times, =)$, que nous abrégeons en K. Pour plus de détails, voir la partie 1.5.

Ici, nous aurons aussi besoin de la structure $(K, +, -, =)$ dans laquelle la multiplication n'est pas autorisée. Ainsi, nos circuits n'auront pas de porte de multiplication,

mais ils auront des portes de sélection pour pouvoir faire des choix.** Il s'agit de portes à trois entrées x, y et z, qui retournent x si $z = 0$ et y sinon : on pourra voir Poizat [65] pour plus de détails. Pour abréger, nous noterons $(K, +)$ cette structure additive. Sur cette structure sont définies les classes $\mathsf{P}_{(K,+)}$ et $\mathsf{NP}_{(K,+)}$. On définit également la classe $\mathsf{NDP}_{(K,+)}$ où la quantification est booléenne : un langage A est dans $\mathsf{NDP}_{(K,+)}$ s'il existe un langage $B \in \mathsf{P}_{(K,+)}$ et un polynôme $p(n)$ tels que

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

Koiran [40] a montré que sur la structure $(K, +)$, la quantification booléenne fournit toute la puissance d'une quantification usuelle : il s'agit du théorème suivant.

3-X – THÉORÈME (Koiran [40])

Sur un corps K de caractéristique nulle,

$$\mathsf{NP}_{(K,+)} = \mathsf{NDP}_{(K,+)}.$$

3.3.2 Idée de la preuve

Nous pouvons maintenant nous consacrer à la preuve du théorème 3-W. Comme dans le théorème 3-E, l'idée est de remplacer la quantification existentielle de $\mathsf{NP}_{(K,+)}$ par un gros produit. Le premier problème est que le domaine de quantification dans $\mathsf{NP}_{(K,+)}$ est infini (c'est K tout entier), alors qu'on effectue des produits sur un domaine fini. Le théorème 3-X permet de se ramener à une quantification sur un domaine fini.

Le second problème que l'on rencontre vient du type de circuits, différent pour Valiant et pour BSS. Dans le dernier cas, on a des portes de test. Or nos gros produits concernent des circuits arithmétiques qui n'en ont pas. Ainsi, il faut éliminer les tests des circuits algébriques de $\mathsf{NP}_{(K,+)}$. Notons que nous n'avions pas ce problème dans le théorème 3-E car les circuits booléens peuvent êtres simulés par des circuits arithmétiques (lemme 1-E).

Pour résoudre ce problème des tests dans les circuits algébriques, on va passer par l'intermédiaire de circuits booléens. Pour cela, il faut manipuler des données booléennes et non l'entrée $\bar{x}$ algébrique. On va donc remplacer $\bar{x}$ par un élément $\bar{q}$ ayant des coordonnées rationnelles de petite taille et qui se comporte exactement de la même façon que $\bar{x}$ dans le circuit algébrique. En encodant ce point rationnel en binaire, les calculs pourront se faire par un circuit booléen, que l'on pourra alors simuler par un circuit arithmétique afin de faire un gros produit. La difficulté est maintenant de trouver un point rationnel $\bar{q}$ de petite taille qui se comporte exactement comme l'entrée $\bar{x}$ dans le circuit algébrique.

Les techniques utilisées concernent la localisation d'un point dans un arrangement d'hyperplans. Des techniques similaires ont été utilisées par Koiran et Fournier dans [27] et [28], mais sur la structure ordonnée $(\mathbb{R}, +, <)$. Ici, on ne s'intéresse qu'à savoir si un point est sur un hyperplan, pas s'il est « au-dessous » ou « au-dessus ».

**De telles portes peuvent être simulées grâce à la multiplication, c'est pourquoi elles ne sont pas présentes habituellement.

Dans la partie 3.3.3, nous expliquerons les notions d'arrangement d'hyperplans et de cellule de l'arrangement. La partie suivante donnera un algorithme pour trouver la cellule d'un point $\bar{x}$. Nous nous en servirons dans la section 3.3.5 pour trouver un point rationnel de petite taille dans la cellule de $\bar{x}$. Enfin, nous pourrons simuler le circuit algébrique sur ce rationnel et conclure.

3.3.3 Arrangement d'hyperplans

Le terme *hyperplan* désignera un hyperplan affine, c'est-à-dire une surface de K^n définie par une équation de la forme $\sum_{i=1}^{n} \lambda_i x_i = \mu$, avec $\lambda_i, \mu \in K$. On emploiera l'adjectif *linéaire* si l'on souhaite préciser que $\mu = 0$. On dira que k hyperplans sont *indépendants* si l'intersection de leurs parties linéaires est de dimension $n-k$. En d'autres termes, ces hyperplans sont en position générale.

Un *arrangement d'hyperplans* est simplement un ensemble fini d'hyperplans $\mathscr{A} = \{H_i;\ i \in I\}$. Un tel ensemble nous permet de définir une relation d'équivalence sur K^n :

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

Les classes d'équivalence sont appelées *cellules* de l'arrangement. Ainsi, deux points sont dans la même cellule s'ils appartiennent exactement aux même hyperplans. Une cellule est donc de la forme

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

où J et J' sont des sous-ensembles de I. On peut supposer sans perte de généralité que les hyperplans $(H_i)_{i \in J}$ sont indépendants. Remarquons au passage que la cellule d'un point $\bar{x} \in K^n$ est caractérisée par un ensemble maximal (au sens de l'inclusion) d'hyperplans indépendants qui contiennent $\bar{x}$. On utilisera cette caractérisation par la suite pour décrire les cellules. Comme on l'a esquissé à la partie 3.3.2, sur l'entrée $\bar{x} \in K^n$, on cherche à déterminer la cellule de $\bar{x}$, c'est-à-dire à renvoyer les indices de ces hyperplans indépendants.

On fixe un polynôme $p(n)$ et on définit $\mathscr{A}_{p,n}$ l'ensemble de tous les hyperplans de K^n à coefficients entiers de valeur absolue majorée par $2^{p(n)}$. On appelle $\mathscr{H}_p$ la famille de tous les arrangements $\mathscr{A}_{p,n}$ (où $n \in \mathbb{N} \setminus \{0\}$). L'intérêt de cette famille provient du lemme suivant.

3-Y – Lemme

Soit $L \in \mathsf{NDP}^0_{(K,+)}$. Il existe un polynôme $p(n)$ tel que pour tous points $\bar{x}$ et $\bar{x}'$ d'une même cellule de $\mathscr{A}_{p,n}$, on ait $\bar{x} \in L$ si et seulement si $\bar{x}' \in L$.

Preuve Par définition, il existe un langage $A \in \mathsf{P}^0_{(K,+)}$ tel que

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

Soit (C_n) une famille de circuits sans constante de taille $r(n)$ polynomiale qui décide A. Puisque la taille de C_n est $r(n)$ et que le circuit ne contient pas de multiplication, tous les tests effectués sur l'entrée $(\bar{x}, \bar{y})$ sont de la forme $\sum_i \lambda_i x_i = \sum_i \mu_i y_i + \gamma$,

où λ_i, μ_i et γ sont des entiers de valeur absolue majorée par $2^{r(n)}$. Si $\bar{y}$ est un uple booléen, le membre droit de l'égalité est majoré en valeur absolue par $(n+1)2^{r(n)}$. Soit $p(n)$ un polynôme tel que $(n+1)2^{r(n)} \leqslant 2^{p(n)}$.

Si $\bar{x}$ et $\bar{x}'$ sont dans la même cellule de $\mathscr{A}_{p,n}$, pour tout uple booléen $\bar{y}$ tous les tests sur les deux couples $(\bar{x}, \bar{y})$ et $(\bar{x}', \bar{y})$ donneront exactement les mêmes résultats. Ainsi, $(\bar{x}, \bar{y})$ et $(\bar{x}', \bar{y})$ seront simultanément dans A ou simultanément hors de A. Donc $\bar{x}$ et $\bar{x}'$ seront simultanément dans L ou simultanément hors de L. ∎

Le but du jeu est maintenant de construire un algorithme qui, sur l'entrée $\bar{x} \in K^n$, retourne la cellule de $\bar{x}$ dans l'arrangement $\mathscr{A}_{p,n}$. On appelle cela la « localisation du point $\bar{x}$ dans l'arrangement ». On aura le droit pour cela de tester si des familles VIIP^0 sont nulles. La définition d'un algorithme avec tests VIIP^0 et la description de l'algorithme de localisation de point font l'objet de la partie suivante.

3.3.4 Localisation d'un point

Dans cette partie, l'objectif est de construire un algorithme avec tests VIIP^0 permettant de trouver la cellule de son entrée $\bar{x} \in K^n$. Nous définissons tout d'abord la notion d'algorithme avec tests VIIP^0, puis nous montrerons qu'on peut résoudre efficacement le problème de la localisation dans un arrangement d'hyperplans grâce aux tests VIIP^0.

3-Z – Définition

Une famille uniforme de circuits algébriques avec tests VIIP^0 est la donnée d'une famille $(f_n(\bar{x})) \in \mathsf{Uniform}\ \mathsf{VIIP}^0$ et d'une famille uniforme de circuits algébriques (C_n), munis de portes de type « $f_n(\bar{y}) = 0\,?$ » (l'indice n est le même pour C_n et f_n). Ces portes sont de degré entrant $|\bar{y}|$ et retournent la valeur 0 si le test échoue (c'est-à-dire si $f_n(\bar{y}) \neq 0$) et 1 sinon.

La classe $\mathsf{P}_K(\mathsf{VIIP}^0)$ est l'ensemble des langages sur K reconnus par une famille uniforme de circuits algébriques de taille polynomiale avec tests VIIP^0.

On voit facilement qu'on obtient la même définition en autorisant un nombre polynomial de familles VIIP^0. En effet, il suffit d'encoder toutes ces familles dans un gros produit et d'ajouter des variables de sélection. Dans les preuves qui suivent, nous utiliserons deux familles VIIP^0 : l'une pour effectuer la localisation, l'autre pour décider un problème NP. Dans un premier temps, nous expliquons comment effectuer la localisation d'un point dans un arrangement d'hyperplans.

3-AA – Proposition

Soit $(\mathscr{H}_p)$ la famille d'arrangements d'hyperplans dont les coefficients sont des entiers bornés par $2^{p(n)}$ en valeur absolue (cette famille a été définie à la section 3.3.3). Il existe une famille uniforme de circuits algébriques de taille polynomiale avec tests VIIP^0 qui, sur l'entrée $\bar{x} \in K^n$, retourne les indices de m hyperplans indépendants qui caractérisent la cellule de $\bar{x}$.

Preuve L'idée de l'algorithme est simple : on maintient un espace de recherche E qui localise $\bar{x}$ le plus précisément possible. Au départ, on n'a aucune information, donc on définit $E = K^n$. Puis à chaque étape on trouve (s'il existe) le premier hyperplan

H de l'arrangement qui raffine E, c'est-à-dire tel que $\bar{x} \in H$ et $\dim(E \cap H) < \dim E$. Puisque la dimension baisse à chaque étape, n étapes sont suffisantes pour trouver la description de la cellule de $\bar{x}$ (c'est-à-dire la liste des indices des hyperplans indépendants que l'on a trouvé durant le processus). Avant d'expliquer comment trouver à chaque étape le premier hyperplan H qui raffine E, résumons l'algorithme :

- $E \leftarrow K^n$;
- $L \leftarrow \emptyset$;
- $R \leftarrow \{H \in \mathscr{A} : \bar{x} \in H\}$;
- tant que $R \neq \emptyset$, faire
  - . soit H_0 le premier hyperplan de R
  - . $L \leftarrow L \cup \{H_0\}$
  - . $E \leftarrow E \cap H_0$
  - . $R \leftarrow \{H \in \mathscr{A} : \bar{x} \in H \text{ et } E \cap H \neq E\}$
- renvoyer L.

Remarquons que $E = \cap_{H \in L} H$ donc conserver la liste L des indices des hyperplans suffit à déterminer E.

Pour trouver le premier hyperplan qui raffine E (noté H_0 dans l'algorithme), on utilise une recherche dichotomique avec des tests VIIP^0. La liste L contient au plus n indices d'hyperplans, tous de taille polynomiale en n. Cette liste sera stockée grâce à un nombre polynomial de variables $l_1, \ldots, l_{q(n)}$ qui représentent le codage booléen des indices d'hyperplans.

À chaque étape, soit A l'ensemble des indices des hyperplans qui ne contiennent pas E. Si f_i est l'équation de H_i, le polynôme

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

s'annule si et seulement si le premier hyperplan qui raffine E a un indice inférieur à j. En faisant varier j, on trouve cet hyperplan en un nombre d'étapes logarithmique en le nombre d'hyperplans, soit polynomial en n.

Il suffit maintenant d'expliquer pourquoi ce produit est dans $\mathsf{Uniform\ VIIP}^0$. Sur l'entrée booléenne $l_1, \ldots, l_{q(n)}$ et i, on peut calculer l'équation de H_i (car l'arrangement $\mathscr{H}_p$ est très simple) et tester si H_i a une trace non-triviale sur E par un simple calcul de rang. Une famille uniforme de circuits de taille polynomiale effectuant une élimination de Gauss permet de réaliser ce calcul. Par le lemme 3-D, on en déduit que ce produit est dans $\mathsf{Uniform\ VIIP}^0$.

Ainsi, en un nombre polynomial de tests VIIP^0, on trouve le premier hyperplan qui raffine E et on peut alors passer à l'étape suivante. Il faut moins de n étapes pour déterminer la cellule de $\bar{x}$: on renvoie la liste L des hyperplans successifs que l'on a trouvés. ∎

3.3.5 Points rationnels de petite taille

Comme on l'a déjà dit, on souhaite travailler avec des objets booléens plutôt qu'algébriques, afin de pouvoir passer par des circuits booléens. C'est pourquoi nous voulons

remplacer l'entrée algébrique $\bar{x}$ par un rationnel $\bar{q}$ de petite taille. Pour cela, $\bar{q}$ doit être dans la cellule de $\bar{x}$: nous cherchons donc à construire un point rationnel de petite taille dans une cellule dont la description est donnée en entrée. L'idée est de supposer d'abord que les hyperplans en jeu sont « simples » (équation de la forme $x_i = 0$), puis de résoudre le cas général en utilisant une application linéaire qui envoie les hyperplans simples sur les vrais hyperplans.

On dit qu'un nombre rationnel est de taille $\leqslant k$ si le numérateur et le dénominateur sont tous deux de valeur absolue majorée par 2^k. Le premier lemme est très simple, il s'agit juste d'évaluer la taille du résultat d'une addition et d'une multiplication de nombres rationnels.

3-AB – Lemme

Soient α et β deux nombres rationnels de taille $\leqslant t$ et $\leqslant t'$ respectivement. Alors
- $\alpha\beta$ est de taille $\leqslant t + t'$;
- $\alpha + \beta$ est de taille $\leqslant t + t' + 1$.

En particulier, si M est une matrice de taille $n \times m$ dont les coefficients sont des rationnels de taille $\leqslant t$, et si x est un vecteur de taille n dont les coefficients sont des rationnels de taille $\leqslant t'$, alors Ax est un vecteur de $\mathbb{Q}^m$ dont les coefficients sont des rationnels de taille $\leqslant n(t + t') + n - 1$.

Afin de trouver un point dans la même cellule que $\bar{x}$, il faut évidemment qu'il soit dans les mêmes hyperplans, mais également hors des mêmes hyperplans. C'est pourquoi on a besoin du lemme suivant qui fournit un point hors d'une famille d'hyperplans.

3-AC – Lemme

Soit $\mathscr{A}$ une famille d'hyperplans dont les coefficients sont des entiers majorés par k en valeur absolue. Alors le point $\bar{q}$ de coordonnées $q_i = (k+1)^i$ (pour $i = 1, \ldots, n$) n'appartient à aucun des hyperplans de $\mathscr{A}$.

Preuve Soit $f(\bar{x}) = \sum_{i=1}^n \alpha_i x_i + b$ l'équation d'un hyperplan H de $\mathscr{A}$. Pour $a \in \mathbb{Z}$, soit $a^+ = \max(0, a)$ et $a^- = \max(0, -a)$. Remarquons que $a = a^+ - a^-$ et $0 \leqslant a^-, a^+ \leqslant k$. Soient $f^+(\bar{x}) = \sum_{i=1}^n \alpha_i^+ x_i + b^+$ et $f^-(\bar{x}) = \sum_{i=1}^n \alpha_i^- x_i + b^-$.

Alors $\bar{q}$ est dans H si et seulement si $f^-(\bar{q}) = f^+(\bar{q})$, c'est-à-dire $\sum_i \alpha_i^-(k+1)^i + b^+ = \sum_i \alpha_i^+(k+1)^i + b^-$. Par unicité de la décomposition en base $(k+1)$, c'est équivalent aux conditions : $b^+ = b^-$ et $\forall i, \alpha_i^- = \alpha_i^+$. Ainsi, $b = 0$ et $\alpha_i = 0$ pour tout i. C'est en contradiction avec l'hypothèse que H est un hyperplan. ∎

On peut maintenant construire un point rationnel de petite taille dans une cellule donnée en entrée. On rappelle que $\mathscr{H}_p$, définie à la section 3.3.3, est la famille des arrangements d'hyperplans dont les coefficients sont des entiers majorés par $2^{p(n)}$ en valeur absolue.

3-AD – Lemme

Pour la famille d'arrangements $\mathscr{H}_p$, il existe une famille uniforme (C_n) de circuits booléens de taille polynomiale en n qui satisfait la propriété suivante : C_n prend en entrée les indices de $m \leqslant n$ hyperplans indépendants de K^n et renvoie un vecteur $\bar{q}$

tel que :
- $\bar{q}$ est dans la cellule définie par les m hyperplans ;
- les coordonnées de $\bar{q}$ sont des rationnels de taille polynomiale en n.

Preuve On ne donnera pas tous les détails de la preuve pour éviter des calculs simples mais fastidieux. L'idée est « d'aplatir » la cellule, c'est-à-dire de la plonger dans K^{n-m}, de trouver un point convenable dans K^{n-m}, puis de remonter dans la cellule de départ.

Soit E l'intersection des m hyperplans : c'est un espace affine de K^n de dimension $n-m$. La cellule est de la forme $E \setminus U$ où U est une union finie (éventuellement vide) d'espaces affines de dimension $n-m-1$. L'équation de E est de la forme $Ax = b$ pour une certaine matrice A de taille $m \times n$. En temps polynomial, on peut trouver m colonnes de A linéairement indépendantes. Supposons pour simplifier qu'il s'agisse des m premières. Soit $\phi : K^{n-m} \to E$ l'application affine qui envoie $(x_{m+1}, \ldots, x_n)$ sur $(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$, où $(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$ est l'unique point de E dont les $n-m$ dernières coordonnées sont $(x_{m+1}, \ldots, x_n)$. La partie linéaire de ϕ est un isomorphisme d'espaces linéaires. Les coefficients de ϕ sont obtenus à partir de ceux de A et b en résolvant un système d'équations linéaires. Il sont donc des nombres rationnels de taille polynomiale en n. Si H est un hyperplan de notre arrangement dont l'intersection avec E est non-triviale, alors $\phi^{-1}(E \cap H)$ est un hyperplan de K^{n-m} dont les coefficients sont des entiers de taille polynomiale en n.

De plus, par le lemme 3-AC on peut construire un point $\bar{q} \in K^{n-m}$ dont les coefficients sont des entiers de taille polynomiale en n et qui sont hors de tous les $\phi^{-1}(E \cap H)$ (pour H dans notre arrangement). Maintenant, par le lemme 3-AB, $\phi(\bar{q})$ a des coefficients rationnels de taille polynomiale en n et il est dans la cellule. ∎

3.3.6 *Décider des problèmes NP*

Nous somme maintenant prêts pour le théorème principal de ce chapitre : les problèmes $\mathsf{NP}_{(K,+)}$ sont décidés par des circuits algébriques de taille polynomiale avec des tests $\mathsf{V\Pi P}^0$.

3-AE – THÉORÈME

Soit K un corps de caractéristique nulle. Alors

$$\mathsf{NP}_{(K,+)} \subseteq \mathsf{P}_K(\mathsf{V\Pi P}^0).$$

Si les produits de taille exponentielle sont calculables par des circuits de taille polynomiale, alors $\mathsf{P}_K(\mathsf{V\Pi P}^0) = \mathsf{P}_K$. Le théorème 3-W est donc un corollaire immédiat du théorème 3-AE.

Preuve du théorème 3-AE Commençons par donner les grandes lignes de la preuve. Tout d'abord, on détermine la cellule de l'entrée $\bar{x}$ puis on construit un point rationnel $\bar{q}$ dans cette cellule. Décider si $\bar{q}$ est dans le langage est un problème NP classique : il peut être décidé par un test $\mathsf{V\Pi P}^0$, ce qui conclut. On peut maintenant donner les petites lignes de la preuve.

Régler le sort des constantes. Remarquons que les classes de Valiant que nous utilisons sont sans constante alors que les classes BSS peuvent utiliser des constantes arbitraires. On peut résoudre ce problème simplement. Soit L un langage de $\mathsf{NP}_{(K,+)}$ qui est une classe avec constantes. Si l'on considère les constantes comme de nouvelles variables (c'est-à-dire que nous faisons comme si elles faisaient partie de l'entrée $\bar{x}$), on obtient un nouveau langage $L' \in \mathsf{NP}^0_{(K,+)}$. Notre construction fournit alors un nouveau circuit avec les même variables en entrée (et avec des tests $\mathsf{V\Pi P}^0$). Il ne reste plus qu'à remplacer les nouvelles variables par les constantes originales pour reconnaître le langage L de départ. Ainsi, dans le reste de la preuve, nous supposerons sans perte de généralité que les classes BSS sont sans constante.

Non-déterminisme booléen. Par le théorème 3-X, nous nous intéressons donc à un langage $L \in \mathsf{NDP}^0_{(K,+)}$. Ainsi, par le lemme 3-Y, il existe un polynôme $p(n)$ tel que pour deux points $\bar{x}$ et $\bar{q}$ dans la même cellule de l'arrangement $\mathscr{A}_{p,n}$, on ait $\bar{x} \in L$ si et seulement si $\bar{q} \in L$.

Trouver la cellule de $\bar{x}$. On applique la proposition 3-AA : il existe une famille uniforme de circuits de taille polynomiale avec tests $\mathsf{V\Pi P}^0$ qui retourne les indices de m hyperplans indépendants caractérisant la cellule de $\bar{x}$.

Trouver un point rationnel dans la cellule. Par le lemme 3-AD, on peut construire en temps polynomial un point $\bar{q}$ dans la cellule de $\bar{x}$, dont les coordonnées sont des rationnels de taille polynomiale. Ainsi, $\bar{x} \in L$ si et seulement si $\bar{q} \in L$.

Décider si un point à coordonnées rationnelles appartient à L est un problème NP. La preuve du théorème 3-E montre alors qu'on peut décider si $\bar{q} \in L$ par un test $\mathsf{V\Pi P}^0$ supplémentaire. ∎

En combinant avec le corollaire 3-V, on obtient le corollaire suivant (mentionné dans l'introduction) qui fait le lien avec les classes de Valiant plus connues.

3-AF – Corollaire

Si Uniform $\mathsf{VP}^0 =$ Uniform VNP^0 alors $\mathsf{NP}_{(K,+)} \subseteq \mathsf{P}_K$.

Ce théorème de transfert ne permet malheureusement pas de prendre en compte la multiplication dans NP_K. En effet, les techniques utilisées manipulent seulement des hyperplans, alors qu'il faudrait manipuler des hypersurfaces. Il existe des algorithmes qui manipulent des hypersurfaces (ou plus précisément des conditions de signe comme on le verra, par exemple ceux de Renegar [67] et de Fichtas, Galligo et Morgenstern [25]), mais malheureusement ils fonctionnent en espace polynomial et pas en temps polynomial. C'est pourquoi nous aurons besoin d'introduire une classe de polynômes dont les coefficients sont calculables en espace polynomial. Mais chut, tout cela sera expliqué dans les chapitres suivants…

CHAPITRE
QUATRE

La classe VPSPACE

QU'EN est-il de la multiplication dans les structures algébriques que nous manipulons ? Contrairement au chapitre précédent, nous ne pouvons plus ignorer cette question. Pour cela, comme nous l'avons déjà mentionné, nous introduisons une nouvelle classe de complexité dans le modèle de Valiant, que nous appelons VPSPACE. En quatorze mots, il s'agit de familles de polynômes dont les coefficients sont calculables en espace polynomial. Nous verrons que cette classe ressemble dans le modèle BSS à la classe PAR des langages décidables par circuits algébriques de profondeur polynomiale, ce qui nous permettra dans les chapitres 5 et 6 d'obtenir deux théorèmes de transfert concernant ces deux classes, l'un sur les complexes et l'autre sur les réels.

Dans ce chapitre, nous nous contentons de définir précisément la classe VPSPACE et quelques variantes (notamment la version uniforme, qui sera celle dont nous nous servirons). Nous montrerons quelques propriétés de cette classe et donnerons un exemple de famille VPSPACE, le résultant d'un système de polynômes. Bien entendu, une question fondamentale est de savoir si cette nouvelle classe peut être calculée par des circuits de taille polynomiale. Et bien entendu, nous ne connaissons pas la réponse. Nous aborderons quand même ce problème à la partie 4.4.

Travaux précédents. Nous introduisons VPSPACE afin de pouvoir utiliser des algorithmes de géométrie algébrique qui fonctionnent en espace polynomial. En particulier, nous aurons besoin d'algorithmes qui énumèrent les conditions de signe d'un ensemble fini de polynômes (les détails seront donnés aux chapitres 5 et 6). Comme nous le verrons, sur $\mathbb{C}$, il s'agit de l'algorithme de Fichtas, Galligo et Morgenstern [25] ; sur $\mathbb{R}$, on utilisera l'algorithme de Renegar [67].
Travaux connexes. Notons que Poizat [66] a défini de manière indépendante une classe de complexité qu'il appelle $\mathsf{VSP}_{\mathsf{dl}}$ et qui s'avère être égale à notre classe VPSPACE. Les circuits qu'il considère contiennent des portes de sommation exponentielle. Nous présenterons brièvement ses travaux à la section 4.2.1.

Plan du chapitre. Dans la première partie, nous définissons la classe VPSPACE ainsi que quelques variantes (uniforme, avec ou sans constantes). La section 4.2 en donne quelques

propriétés, notamment une caractérisation en terme de circuits arithmétiques de profondeur polynomiale. Nous verrons à la partie 4.3 un exemple de famille VPSPACE sur $\mathbb{C}$: le résultant d'un système de polynômes.

À la partie 4.4, nous étudions la question de savoir si VPSPACE admet des circuits de taille polynomiale. La première approche consiste à donner des hypothèses équivalentes en termes de classes plus connues, comme P, PSPACE, VP et VNP. La seconde approche se place du point de vue de la complexité booléenne et montre qu'une hypothèse de complexité de Kolmogorov (la symétrie de l'information en espace polylogarithmique) permet de séparer VPSPACE et $\mathsf{VP}_{\mathsf{nb}}$. Plus de détails sur le contexte sont donnés au début de cette partie.

Ce chapitre reprend une partie de l'article [48], dans lequel la classe VPSPACE apparaît pour la première fois, et s'inspire de l'article [64] en ce qui concerne la seconde moitié de la partie 4.4.

4.1 Définition

Nous fixons un corps K arbitraire, sur lequel nos polynômes seront définis. Ce corps sera implicite dans nos notations. Pour les résultats de transfert des prochains chapitres, nous travaillerons sur $\mathbb{R}$ ou $\mathbb{C}$, mais le choix de tout autre corps de caractéristique nulle pour VPSPACE ferait l'affaire car on utilisera une version sans constante (notons toutefois que le choix du corps aura son importance pour la classe PAR du modèle BSS).

La définition de VPSPACE est donnée en termes de *fonction coefficient*, notion que nous définissons d'abord. On rappelle qu'un monôme $x_1^{\alpha_1}\cdots x_{u(n)}^{\alpha_{u(n)}}$ est encodé en binaire par $\alpha=(\alpha_1,\ldots,\alpha_{u(n)})$ et on l'écrira $\bar{x}^{\alpha}$.

4-A – Définition

Soit (f_n) une famille de polynômes à plusieurs variables et à coefficients entiers. La fonction coefficient de (f_n) est la fonction a qui retourne, sur l'entrée (n,α,i), le i-ème bit $a(n,\alpha,i)$ du coefficient du monôme $\bar{x}^{\alpha}$ dans f_n. De plus, $a(n,\alpha,0)$ est le signe du coefficient du monôme $\bar{x}^{\alpha}$. Ainsi, f_n s'écrit

$$f_n(\bar{x})=\sum_{\alpha}\Big((-1)^{a(n,\alpha,0)}\sum_{i\geqslant 1}a(n,\alpha,i)2^{i-1}\bar{x}^{\alpha}\Big).$$

La fonction coefficient est une fonction $a:\{0,1\}^*\to\{0,1\}$ et peut donc être vue comme un langage. Cela nous permet de parler aisément de la complexité d'une fonction coefficient. Nous pouvons maintenant passer à la définition de la classe VPSPACE. Dans un premier temps, nous définissons une version sans constante ; il sera facile de les ajouter par la suite.

4-B – Définition

Une famille (f_n) de polynômes $f_n\in K[x_1,\ldots,x_{u(n)}]$ est dans la classe $\mathsf{VPSPACE}^0$ si elle satisfait les propriétés suivantes :

1. le nombre de variables $u(n)$ est polynomialement borné ;
2. les polynômes f_n sont à coefficients entiers ;

3. la taille des coefficients de f_n est bornée par $2^{p(n)}$ pour un certain polynôme p ;
4. le degré de f_n est borné par $2^{p(n)}$ pour un certain polynôme p ;
5. la fonction coefficient de (f_n) est dans PSPACE/poly.

La version avec constantes suit immédiatement.

4-C – Définition

Soit (f_n) une famille de polynômes à coefficients dans K. La famille (f_n) est dans VPSPACE s'il existe $(g_n(\bar{x},\bar{y})) \in \mathsf{VPSPACE}^0$ et une famille de constantes $(\bar{\alpha}^{(n)})$ tels que $f_n(\bar{x}) = g_n(\bar{x},\bar{\alpha}^{(n)})$.

4-D – Remarque La non-uniformité ici consiste en un conseil de taille polynomiale pour calculer la fonction coefficient. Comme on le verra plus tard, les polynômes VPSPACE sont calculés par des circuits de profondeur polynomiale, de taille éventuellement exponentielle : les circuits ne peuvent alors pas être arbitraires (car la non-uniformité serait « exponentielle »), il doivent être construits par un algorithme P/poly.

Comme nous l'avons déjà mentionné plusieurs fois, dans le cadre de cette étude nous nous concentrons sur des classes uniformes. Ainsi, nous travaillerons plutôt avec la classe Uniform VPSPACE^0 que l'on définit maintenant. Il semblait plus cohérent toutefois de commencer par la version non-uniforme car il en est ainsi dans la tradition de Valiant et c'est également la définition originale de [48].

4-E – Définition

La classe Uniform VPSPACE^0 suit la définition 4-B mais dans laquelle la fonction coefficient du point 5 doit être dans PSPACE (c'est-à-dire uniforme).

Passons maintenant à l'étude de la classe Uniform VPSPACE^0 ou de ses variantes.

4.2 Quelques propriétés

Nous commençons par voir deux autres caractérisations de VPSPACE, puis étudierons quelques propriétés de clôture.

4.2.1 Autres caractérisations

À la fin de cette partie, nous mentionnerons les résultats de Poizat [66] sur une formulation équivalente de VPSPACE par des circuits possédant des portes de sommation exponentielle. Mais tout d'abord nous montrons que les familles Uniform VPSPACE^0 sont exactement celles calculées par une famille uniforme de circuits arithmétiques sans constante et de profondeur polynomiale. L'uniformité est ici polynomiale, mais on pourrait utiliser de manière équivalente l'uniformité L ou PSPACE, voir pour cela la remarque 4-H. Pour faciliter la compréhension, nous introduisons temporairement la classe Uniform VPAR^0, que nous montrerons être égale à Uniform VPSPACE^0.

4-F – Définition

La classe Uniform VPAR^0 est l'ensemble des familles de polynômes (f_n), avec un nombre polynomial de variables, calculées par une famille uniforme de circuits arithmétiques sans constante de profondeur polynomiale. Notons que les circuits peuvent ainsi être de taille exponentielle.

4-G – Proposition

Sur un corps K quelconque, Uniform $\mathsf{VPSPACE}^0 =$ Uniform VPAR^0.

Preuve Soit (f_n) une famille Uniform $\mathsf{VPSPACE}^0$. Afin de calculer f_n par un circuit de profondeur polynomiale, on calcule d'abord en parallèle tous ses monômes, ce que l'on peut faire en profondeur polynomiale grâce au lemme 1-L puisque la fonction coefficient de (f_n) est dans PSPACE. Puis il suffit de les ajouter tous, ce que l'on peut faire en profondeur polynomiale en suivant un arbre binaire complet. Le circuit ainsi créé a une profondeur polynomiale et est uniforme car la fonction coefficient de (f_n) l'est.

Réciproquement, considérons un circuit arithmétique de profondeur polynomiale. Nous montrons que l'on peut construire un circuit booléen de profondeur polynomiale qui prend en entrée le codage α d'un monôme et calcule le coefficient de $\bar{x}^\alpha$. Nous raisonnons par récurrence en calculant le coefficient de $\bar{x}^\alpha$ pour toute porte du circuit arithmétique. Pour les entrées, il n'y a pas de problème. Pour une porte d'addition $b + c$, il suffit d'ajouter les coefficients des deux portes b et c. Pour une porte de multiplication $b \times c$, on calcule en parallèle

$$\sum_{\beta+\gamma=\alpha} \mathrm{coef}_b(\bar{x}^\beta)\mathrm{coef}_c(\bar{x}^\gamma),$$

où $\mathrm{coef}_b(\bar{x}^\beta)$ est le coefficient du monôme $\bar{x}^\beta$ dans la porte b et similairement pour $\mathrm{coef}_c(\bar{x}^\gamma)$. Puisque l'on sait calculer cette somme de termes (tous de taille simplement exponentielle) en profondeur polynomiale, le circuit booléen ainsi construit est uniforme et de profondeur polynomiale. Ainsi, par le lemme 1-L, la fonction coefficient de (f_n) est dans PSPACE. ∎

4-H – Remarque En utilisant le lemme 1-L, cette preuve fonctionne de la même façon avec les uniformités L ou PSPACE plutôt que P, ce qui montre que ces trois notions définissent la même classe uniforme.

Ce lemme montre les ressemblances de la classe Uniform $\mathsf{VPSPACE}^0$ avec la classe PAR^0_K du modèle BSS, consistant en l'ensemble des langages sur K reconnus par une famille uniforme de circuits algébriques de profondeur polynomiale. Mais bien entendu, les tests sont autorisés pour PAR^0_K et pas pour Uniform $\mathsf{VPSPACE}^0$. Les théorèmes de transferts des chapitres 5 et 6 préciserons le lien entre les deux classes.

4-I – Remarque Ce résultat est ici énoncé dans un contexte uniforme et sans constante. Il est clair qu'il peut être généralisé aux autres contextes en considérant les versions appropriées de la classe VPAR (notamment, les constantes éventuelles doivent être en nombre polynomial).

Comme promis, nous exposons maintenant la vision de Poizat [66] sur VPSPACE. Poizat introduit les circuits et la classe de complexité suivants. Nous renvoyons à l'article original [66] pour plus de précisions car des détails techniques sont omis ici pour plus de clarté.

4-J – Définition

> Un *S-circuit* est un circuit arithmétique muni de portes de sommation, c'est-à-dire de portes dont l'entrée est un polynôme $f(\bar{x},\bar{y})$ et dont la sortie est le polynôme $g(\bar{x}) = \sum_{\bar{\epsilon}\in\{0,1\}^{|\bar{y}|}} f(\bar{x},\bar{\epsilon})$.
> La classe $\mathsf{VSP}_{\mathsf{dl}}$ est l'ensemble des familles de polynômes calculables par une famille de S-circuits de taille polynomiale.

Ainsi, il s'agit d'une extension assez naturelle de VNP, dans laquelle la somme exponentielle n'est pas reléguée à la fin mais plusieurs sommes peuvent être calculées directement dans le circuit. D'ailleurs, quand on se restreint à des circuits multiplicativement disjoints, on retrouve la classe VNP. Mais l'intérêt de cette classe pour ce chapitre réside dans le résultat suivant de [66].

4-K – Proposition

> La classe $\mathsf{VSP}_{\mathsf{dl}}$ est égale à VPSPACE.

4-L – Remarque Poizat [66] montre également que les portes de sommation exponentielle peuvent être remplacées par des portes d'évaluation en 0 et en 1. Ainsi, la classe $\mathsf{VSP}_{\mathsf{dl}}$ n'est pas sans ressemblance avec la caractérisation de PSPACE par Babai et Fortnow [3]. Mentionnons également, même si ceci ne nous servira pas ici, que l'un des résultat principaux de [66] est la construction d'un S-circuit de taille linéaire pour calculer la factorielle.

4.2.2 Quelques propriétés de clôture

En utilisant la caractérisation de la proposition 4-G, il est facile de montrer le lemme suivant.

4-M – Lemme

> La classe Uniform $\mathsf{VPSPACE}^0$ est close par produits et sommes de taille exponentielle.

Toujours par la caractérisation de la proposition 4-G, il est clair que VPSPACE contient $\mathsf{VP}_{\mathsf{nb}}$. Le résultat suivant découle alors du lemme précédent.

4-N – Lemme

> Les classes $\mathsf{VP}_{\mathsf{nb}}$, $\mathsf{VNP}_{\mathsf{nb}}$ et VIIP sont incluses dans VPSPACE.

En réalité, on peut même effectuer des sommes et des produits sur des ensembles plus compliqués que $\{0,1\}^{p(n)}$, comme on va le voir maintenant.

4-O – Lemme

> Soient A un langage dans PSPACE, $(f_n(\bar{x},\bar{y}))$ une famille dans Uniform $\mathsf{VPSPACE}^0$ et $p(n)$ un polynôme, où $|\bar{y}| = p(n)$. Alors les familles $(g_n(\bar{x}))$ et $(h_n(\bar{x}))$ définies

ci-dessous sont dans Uniform VPSPACE^0.

$$g_n(\bar{x}) = \sum_{\bar{\epsilon} \in A^{=p(n)}} f_n(\bar{x}, \bar{\epsilon}) \text{ et } h_n(\bar{x}) = \prod_{\bar{\epsilon} \in A^{=p(n)}} f_n(\bar{x}, \bar{\epsilon}).$$

Preuve Il suffit d'utiliser le lemme 4-M puisque l'on a

$$\sum_{\bar{\epsilon} \in A^{=p(n)}} f_n(\bar{x}, \bar{\epsilon}) = \sum_{\bar{\epsilon} \in \{0,1\}^{p(n)}} \chi_A(\bar{\epsilon}) f_n(\bar{x}, \bar{\epsilon}) \text{ et}$$

$$\prod_{\bar{\epsilon} \in A^{=p(n)}} f_n(\bar{x}, \bar{\epsilon}) = \prod_{\bar{\epsilon} \in \{0,1\}^{p(n)}} [\chi_A(\bar{\epsilon}) f_n(\bar{x}, \bar{\epsilon}) + (1 - \chi_A(\bar{\epsilon}))],$$

où χ_A, la fonction caractéristique de A, est dans Uniform VPSPACE^0 par le lemme 1-E et par la proposition 4-G : en effet, A est reconnu par une famille uniforme de circuits booléens de profondeur polynomiale par le lemme 1-L. ∎

Voyons maintenant un exemple de famille VPSPACE.

4.3 Un exemple

Dans cette partie, nous donnons un exemple de famille Uniform VPSPACE^0 sur $\mathbb{C}$, à savoir le résultant d'un système d'équations polynomiales. Dans un premier temps, nous expliquons ce qu'est un résultant, puis nous montrons qu'on peut effectivement le calculer dans Uniform VPSPACE^0.

Avant d'expliquer la construction d'un résultant[††], on peut déjà mentionner une de ses caractéristiques importantes : un système de $n+1$ équations polynomiales homogènes en $n+1$ variables a une solution non triviale si et seulement si son résultant est nul. On pourra trouver plus de détails sur les résultants dans Macaulay [56] ou Canny [17].

Soit $f_1, \ldots, f_{n+1} \in \mathbb{C}[X_0, \ldots, X_n]$ un système de $n+1$ polynômes homogènes. Le résultant du système est égal au quotient des déterminants de deux matrices M et M' :

$$R = \frac{\det M}{\det M'} \tag{4.1}$$

où les coefficients de M sont parmi ceux des f_i, et où M' est une sous-matrice de M. La matrice M est appelée *matrice de Macaulay* (c'est une généralisation de la matrice de Sylvester pour deux polynômes à une variable) et c'est elle que nous décrivons maintenant. Soient d_i le degré de f_i et $d = 1 + \sum_{i=1}^{n+1}(d_i - 1)$. On notera Mon_d l'ensemble de tous les monômes en $X_0, \ldots, X_n$ de degré d : le cardinal de Mon_d est $N = \binom{d+n}{d}$.

La matrice M a N lignes et N colonnes, les deux étant indexées par les éléments de Mon_d. La ligne correspondant au monôme $\bar{x}^\alpha$ représente alors le polynôme

$$\frac{\bar{x}^\alpha}{x_i^{d_i}} f_i, \text{ où } i = \min\{j ; x_j^{d_j} \text{ divise } \bar{x}^\alpha\}.$$

[††]Nous aborderons en effet les résultants seulement du point de vue de leur construction.

Enfin, la sous-matrice M' consiste en les lignes et colonnes de M qui ne sont pas « réduites », voir Canny [17] pour plus de détails. Ce que nous allons calculer ici n'est pas le résultant R lui-même, mais plutôt un multiple d'icelui : le numérateur $\det M$. Lorsque $\det M' \neq 0$, cela ne change rien si nous nous intéressons seulement à savoir si R s'annule. Notons par ailleurs qu'on est aussi capable de calculer le dénominateur $\det M'$.

Pour simplifier, nous supposerons que tous les d_i sont égaux, de valeur commune δ. Le nombre n de polynômes et de variables tendra vers l'infini, mais le paramètre δ restera constant. On encodera un système $(f_1, \ldots, f_{n+1})$ de $n+1$ polynômes homogènes en $n+1$ variables et de degré δ par la liste des coefficients des polynômes, c'est-à-dire par $k(n+1)$ variables $(a_{1,1}, \ldots, a_{1,k}, a_{2,1}, \ldots, a_{n+1,k})$ où $k = \binom{n+\delta}{\delta}$ est le nombre de monômes de degré δ en $n+1$ variables. Remarquons que k est polynomial en n pour tout δ fixé.

On définit alors $\mathrm{Mac}_n^\delta(f_1, \ldots, f_{n+1})$ égal à la matrice de Macaulay M de $(f_1, \ldots, f_{n+1})$. Cette matrice est carrée de taille $\binom{n+d}{d}$, où $d = 1+(n+1)(\delta-1)$. Bien entendu, cette taille est exponentielle en n dès que $\delta \geqslant 2$. Mais calculer le déterminant de M se fait par un circuit de profondeur polylogarithmique en la taille de M, donc polynomiale en n (voir Berkowitz [6] pour le calcul du déterminant en parallèle, ou plus généralement Valiant, Skyum, Berkowitz et Rackoff [82] pour un lemme de parallélisation). Ces considérations prouvent la proposition suivante.

4-P – Proposition

> Pour tout δ fixé, la famille $(\det(\mathrm{Mac}_n^\delta))$ (le déterminant de la matrice de Macaulay d'un système de $n+1$ polynômes homogènes de degré δ en $n+1$ variables) se trouve dans la classe Uniform $\mathsf{VPSPACE}^0$.

De la même façon, la famille des déterminants des matrices M' dans l'équation (4.1) (qui forment les dénominateurs des résultants) est dans Uniform $\mathsf{VPSPACE}^0$.

4.4 VPSPACE admet-elle de petits circuits ?

La classe VPSPACE que nous venons de définir semble ainsi être assez robuste. En effet, elle admet plusieurs caractérisations intéressantes (en terme de fonction coefficient, de circuits de profondeur polynomiale ou encore de circuits avec portes de sommation) et est de plus close par sommes et produits de taille exponentielle. Par ailleurs, nous avons donné un exemple naturel d'une famille de polynômes venant de géométrie algébrique, le résultant, qui se trouve dans cette classe.

VPSPACE contient toutes les classes de Valiant qui existent jusqu'à présent et le premier réflexe est bien sûr de savoir si ces inclusions sont strictes. Ne soyons point trop ambitieux, le but du jeu est seulement de voir si l'on peut séparer la plus petite, $\mathsf{VP}_{\mathsf{nb}}$, de la plus grande, VPSPACE. En d'autres termes, il s'agit de savoir si VPSPACE admet des circuits de taille polynomiale. Comme il est de coutume en complexité, nous ne savons pas répondre à cette question.

Nous allons d'abord donner des hypothèses équivalentes qui montrent que la séparation est hautement probable. Afin de clarifier la situation de VPSPACE, l'idée est de trouver une hypothèse faisant intervenir des classes connues et qui soit équivalente à $\mathsf{VP}_{\mathsf{nb}} = \mathsf{VPSPACE}$. Puis nous donnerons une hypothèse venant de complexité de Kol-

mogorov qui implique la séparation. Nous ne travaillerons alors que du côté de la complexité booléenne.

Plan de cette partie. Dans la première section, nous travaillerons en complexité algébrique autour de l'hypothèse $\mathsf{VP}_{\mathsf{nb}} = \mathsf{VPSPACE}$. Nous verrons ce qu'elle implique et ce qui l'implique. Dans la seconde section, le but sera de séparer les version non-uniformes de PSPACE et NC. En effet, si VPSPACE = VP alors ces deux classes coïncideraient, c'est pourquoi nous tentons de les séparer. Nous y arriverons seulement sous une hypothèse issue de la complexité de Kolmogorov en ressources bornées. Puisque les deux sections sont assez différentes, nous avons écrit une introduction et des commentaires au début de chacune.

La première section reprend des résultats de l'article [48] et la seconde s'appuie sur une partie de [64].

4.4.1 Hypothèses équivalentes

Comme mentionné ci-dessus, la classe VPSPACE est grande (elle contient notamment toutes les classes de Valiant étudiées jusqu'alors) et vraisemblablement strictement plus que $\mathsf{VP}_{\mathsf{nb}}$. Mais une preuve de leur séparation semble hors de portée pour l'instant. Toutefois, afin de clarifier la situation, nous allons exprimer cette séparation de manière équivalente grâce à des classes de complexité bien connues, notamment VP, VNP, P et PSPACE.

Travaux précédents. Koiran [43] a étudié les hypothèses sous lesquelles on est capable de calculer efficacement certaines suites d'entiers dans le modèle de Valiant. Il a notamment montré que sous l'hypothèse P = PSPACE et VP = VNP, la factorielle $(n!)$ est facile à calculer. C'est typiquement le genre d'hypothèse que nous allons utiliser dans cette partie. Encore plus récemment, Bürgisser [15] a montré que l'hypothèse P = PSPACE n'était pas nécessaire pour le résultat de Koiran. Enfin, pour les résultats que nous allons montrer ici, nous utiliserons notamment une méthode d'élimination des constantes issue de Bürgisser [13].

Plan de cette section. Nous allons voir successivement plusieurs résultats. Le premier sera une caractérisation de la question « $\mathsf{VP}_{\mathsf{nb}} = \mathsf{VPSPACE}$? » en termes des classes VP, VNP, P/poly et PSPACE/poly. Pour cela, nous aurons besoin de définir une version « bornée » de VPSPACE, où le degré et la taille des coefficients seront polynomiaux. Puis nous verrons une version uniforme du résultat précédent. Enfin, on donnera une implication en complexité booléenne de la version uniforme de l'hypothèse que VPSPACE a des circuits de taille polynomiale. Ce dernier résultat suggérera que VPSPACE n'a pas de circuits de taille polynomiale.

Commençons donc par donner une hypothèse équivalente à $\mathsf{VP}_{\mathsf{nb}} = \mathsf{VPSPACE}$. Afin de comparer VP et VNP, qui sont des classes de polynômes de degré polynomialement borné, d'une part, et VPSPACE où le degré peut être exponentiel, d'autre part, il convient de définir temporairement une version « bornée » de VPSPACE. Cela mène à la définition suivante.

4-Q – Définition

Une famille (f_n) de polynômes est dans $\mathsf{VPSPACE}_b^0$ si $(f_n) \in \mathsf{VPSPACE}^0$ et si la taille des coefficients et le degré de f_n sont polynomialement bornés.
De plus, la classe $\mathsf{VPSPACE}_b$ est définie à partir de $\mathsf{VPSPACE}_b^0$ en ajoutant des constantes de la même façon que VPSPACE est définie à partir de $\mathsf{VPSPACE}^0$.

Les deux lemmes suivants justifient l'introduction de $\mathsf{VPSPACE}_b$.

4-R – Lemme

$$\mathsf{VPSPACE}_b = \mathsf{VP} \iff \mathsf{VPSPACE} = \mathsf{VP}_{nb}.$$

Preuve Supposons tout d'abord que $\mathsf{VPSPACE} = \mathsf{VP}_{nb}$ et prenons une famille (f_n) de $\mathsf{VPSPACE}_b$. Puisque $\mathsf{VPSPACE}_b \subset \mathsf{VPSPACE}$, (f_n) est dans VP_{nb} par hypothèse. Mais puisque le degré de (f_n) est polynomialement borné, $(f_n) \in \mathsf{VP}$.

Réciproquement, soit (f_n) une famille de VPSPACE : elle s'écrit $f_n(\bar{x}) = g_n(\bar{x}, \bar{a}^{(n)})$ où $(\bar{a}^{(n)})$ est une famille de constantes et $(g_n(\bar{x}, \bar{y})) \in \mathsf{VPSPACE}^0$. Pour simplifier, renommons les $u(n)$ variables de g_n en $v_1, \ldots, v_{u(n)}$: nous obtenons ainsi

$$g_n(\bar{v}) = \sum_{\alpha} \Big((-1)^{a(n,\alpha,0)} \sum_{i=1}^{2^{p(n)}} a(n,\alpha,i) 2^{i-1} \bar{v}^{\alpha} \Big),$$

où a est dans $\mathsf{PSPACE/poly}$. Dans cette expression, $p(n)$ est un polynôme et $2^{p(n)}$ est un majorant du degré et de la taille des coefficients de g_n. Afin d'utiliser l'hypothèse $\mathsf{VPSPACE}_b = \mathsf{VP}$, il faut définir une famille $(h_n) \in \mathsf{VPSPACE}_b^0$ qui va en quelque sorte « simuler » (g_n). Pour cela, nous définissons

$$\Big(h_n(z_{1,1}, \ldots, z_{1,p(n)}, z_{2,1}, \ldots, z_{u(n),p(n)}, w_1, \ldots, w_{p(n)}) \Big),$$

où, intuitivement, la variable $z_{i,j}$ va remplacer $v_i^{2^j}$ dans g_n et w_i va prendre la valeur 2^{2^i}. Plus formellement, h_n est définie ainsi :

- on remplace v_i^k dans g_n par le produit $\prod_{j \in J_k} z_{i,j}$, où l'ensemble J_k est constitué des bits à 1 dans la représentation binaire de k ;
- dans le terme $\sum_{i=1}^{2^{p(n)}} a(n,\alpha,i) 2^{i-1}$ de g_n, le coefficient 2^{i-1} est remplacé par le produit $\prod_{j \in J_{i-1}} w_j$, où l'ensemble J_{i-1} est constitué des bits à 1 dans la représentation binaire de $i-1$.

Le degré de h_n est alors polynomialement borné et tous les coefficients valent -1, 0 ou 1. De plus, la fonction coefficient est encore dans $\mathsf{PSPACE/poly}$. Ainsi $(h_n) \in \mathsf{VPSPACE}_b^0$, donc $(h_n) \in \mathsf{VP}$ par hypothèse. Il ne reste plus qu'à remplacer $z_{i,j}$ par $v_i^{2^j}$ et w_i par 2^{2^i} pour montrer que $(g_n(\bar{v}) = g_n(\bar{x}, \bar{y})) \in \mathsf{VP}_{nb}$, puis de remplacer $\bar{y}$ par les constantes originales afin de montrer que $(f_n) \in \mathsf{VP}_{nb}$. ∎

4-S – Lemme

$\mathsf{VPSPACE}_b$ contient VNP.

Preuve Soit (HC_n) la famille, déjà vue au chapitre 2, définie par

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

où la somme est prise sur tous les n-cycles σ sur $\{1,\ldots,n\}$. Ce polynôme compte le nombre de cycles hamiltoniens dans un graphe donné par sa matrice d'adjacence. On sait que (HC_n) est VNP-complet, voir Valiant [79] pour la preuve originale ou Malod [57] pour une preuve simplifiée. Puisque $\mathsf{VPSPACE}_\mathsf{b}$ est close par p-projections et contient HC_n, le lemme est prouvé. ∎

On se propose maintenant de montrer l'équivalence suivante, permettant de mieux comprendre la question de savoir si VPSPACE a des circuits de taille polynomiale.

4-T – Proposition

Sous l'hypothèse de Riemann généralisée,

$$\mathsf{VP}_\mathsf{nb} = \mathsf{VPSPACE} \iff [\mathsf{P/poly} = \mathsf{PSPACE/poly} \text{ et } \mathsf{VP} = \mathsf{VNP}].$$

De plus, l'implication de droite à gauche est vraie même sans supposer l'hypothèse de Riemann généralisée.

Preuve Supposons d'abord que P/poly = PSPACE/poly et VP = VNP. Par le lemme 4-R, l'égalité $\mathsf{VPSPACE} = \mathsf{VP}_\mathsf{nb}$ est équivalente à son analogue borné $\mathsf{VPSPACE}_\mathsf{b} = \mathsf{VP}$. Soit $(f_n) \in \mathsf{VPSPACE}_\mathsf{b}$: sa fonction coefficient est dans PSPACE/poly, donc dans P/poly par la première partie de l'hypothèse. Puisque l'ensemble des fonctions coefficient de familles VNP contient $\oplus$P/poly (voir par exemple Bürgisser [13]), donc P/poly, (f_n) est en fait dans VNP. En utilisant l'autre partie de l'hypothèse, (f_n) est dans VP.

Réciproquement, supposons que $\mathsf{VPSPACE} = \mathsf{VP}_\mathsf{nb}$. Encore une fois, c'est équivalent à $\mathsf{VPSPACE}_\mathsf{b} = \mathsf{VP}$ par le lemme 4-R. Ainsi VNP = VP puisque $\mathsf{VP} \subseteq \mathsf{VNP} \subseteq \mathsf{VPSPACE}_\mathsf{b}$ par le lemme 4-S. Il reste à montrer qu'un langage A quelconque dans PSPACE/poly appartient en fait à P/poly. Par le lemme 1-M, A est reconnu par une famille P/poly-uniforme de circuits booléens de profondeur polynomiale. Par le lemme 1-E et la proposition 4-G, il existe une famille $(f_n) \in \mathsf{VPSPACE}$ telle que sur toute entrée booléenne $\bar{x} \in \{0,1\}^n$, on ait $f_n(\bar{x}) \in \{0,1\}$ et $f_n(\bar{x}) = 1$ si et seulement si $\bar{x} \in A$.

Par l'hypothèse, $(f_n) \in \mathsf{VP}_\mathsf{nb}$, donc il existe une famille (C_n) de circuits arithmétiques de taille polynomiale, avec des constantes arbitraires provenant du corps sous-jacent K, qui calcule (f_n). Afin d'évaluer ces circuits sur des entrées booléennes grâce à des circuits booléens, il s'agit maintenant d'éliminer les constantes de K. On procède comme dans Bürgisser [13]. Soient $\bar{y} \in K^{|\bar{y}|}$ les constantes utilisées par le circuit C_n et appelons $g_n(\bar{X},\bar{Y})$ le polynôme calculé par C_n lorsque ces constantes sont remplacées par les nouvelles variables $\bar{Y}$. Ainsi, $g_n(\bar{X},\bar{y}) = f_n(\bar{X})$, donc le système S d'équations en $\bar{Y}$ défini par

$$S = \left(g_n(\bar{x},\bar{Y}) = f_n(\bar{x})\right)_{\bar{x}\in\{0,1\}^n}$$

a une solution $\bar{y}$ sur K, donc en particulier sur sa clôture algébrique $\bar{K}$. Toutes les équations dans ce système ont des coefficients entiers, leur degré est borné par $2^{q(n)}$ et leur poids par $2^{2^{q(n)}}$ pour un certain polynôme q, où le poids d'un polynôme est la somme des valeurs absolues de ses coefficients.

Par le théorème 4.4 de Bürgisser [13, p. 64], sous l'hypothèse de Riemann généralisée il existe un nombre premier $p \leqslant 2^{n^2 q(n)}$ tel que S a une solution sur le corps $\mathbb{Z}/p\mathbb{Z}$. En effet, il existe un tel $p \leqslant a$ dès que

$$\frac{\pi(a)}{d^{O(n)}} > \sqrt{a}\log(wa),$$

où d et w sont des bornes sur le degré et le poids des équations, respectivement, et $\pi(a)$ est le nombre de nombres premiers $\leqslant a$. Ainsi, il existe un circuit arithmétique de taille polynomial sur $\mathbb{Z}/p\mathbb{Z}$ qui calcule le polynôme $g_n(\bar{X}, \bar{y}')$ et ce polynôme prend les mêmes valeurs que $f_n(\bar{X})$ sur les entrées booléennes.

On remarquera que la taille de p est polynomiale et qu'une solution $\bar{y}'$ de ce système S sur $\mathbb{Z}/p\mathbb{Z}$ a également une taille polynomiale. Donc un circuit booléen de taille polynomiale, travaillant modulo p, peut facilement calculer la valeur de $g_n(\bar{X}, \bar{y}')$ sur $\mathbb{Z}/p\mathbb{Z}$. Ce circuit booléen a la même valeur que f_n sur les entrées booléennes. Ainsi, $A \in \mathsf{P/poly}$ et le résultat est montré. ∎

Passons maintenant à la version uniforme et sans constante de l'hypothèse, qui est plus forte que celle de la proposition précédente. Pour la preuve, on aura besoin de se rappeler la définition de degré formel complet et la définition de VP^0 en ces termes (voir le chapitre 1). Voici donc une hypothèse dans un contexte uniforme. Notons d'ores et déjà qu'il n'est pas clair qu'on puisse remplacer l'hypothèse $\mathsf{Uniform\ VP}^0_{\mathsf{nb}} = \mathsf{Uniform\ VNP}^0_{\mathsf{nb}}$ par $\mathsf{Uniform\ VP}^0 = \mathsf{Uniform\ VNP}^0$ car il nous faut simuler des circuits booléens de taille polynomiale arbitraires.

4-U – Proposition

$$\mathsf{Uniform\ VP}^0_{\mathsf{nb}} = \mathsf{Uniform\ VPSPACE}^0 \text{ si et seulement si}$$
$$\mathsf{P} = \mathsf{PSPACE} \text{ et } \mathsf{Uniform\ VP}^0_{\mathsf{nb}} = \mathsf{Uniform\ VNP}^0_{\mathsf{nb}}.$$

Preuve Supposons d'abord que $\mathsf{P} = \mathsf{PSPACE}$ et $\mathsf{Uniform\ VP}^0_{\mathsf{nb}} = \mathsf{Uniform\ VNP}^0_{\mathsf{nb}}$. Soit (f_n) une famille dans $\mathsf{Uniform\ VPSPACE}^0$. Sa fonction coefficient est dans PSPACE, donc dans P par la première partie de l'hypothèse. La somme des monômes avec leur coefficient est donc dans $\mathsf{Uniform\ VNP}^0_{\mathsf{nb}}$. Ainsi, $(f_n) \in \mathsf{Uniform\ VP}^0_{\mathsf{nb}}$ par la seconde partie de l'hypothèse.

Réciproquement, montrons dans un premier temps que $\mathsf{P} = \mathsf{PSPACE}$. Soit A un langage de PSPACE : par le lemme 1-L, il est reconnu par une famille uniforme de circuits booléens de profondeur polynomiale. Par le lemme 1-E et la proposition 4-G, on obtient une famille de polynômes $(f_n) \in \mathsf{Uniform\ VPSPACE}^0$ qui prennent la même valeur sur les entrées booléennes, c'est-à-dire

$$\forall \bar{x} \in \{0,1\}^n, f_n(\bar{x}) \in \{0,1\} \text{ et } [f_n(\bar{x}) = 1 \iff \bar{x} \in A].$$

Par hypothèse, $(f_n) \in \mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$ de sorte qu'il existe une famille uniforme de circuits arithmétiques de taille polynomiale qui calcule (f_n). Bien sûr, de tels circuits peuvent aisément être évalués en temps polynomial sur des entrées booléennes (en travaillant modulo 2 pour éviter les dépassements de capacité). Cela implique que $\mathsf{PSPACE} = \mathsf{P}$.

Par ailleurs, l'égalité $\mathsf{Uniform}\ \mathsf{VNP}^0_{\mathsf{nb}} = \mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$ suit immédiatement des inclusions $\mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}} \subseteq \mathsf{Uniform}\ \mathsf{VNP}^0_{\mathsf{nb}} \subseteq \mathsf{Uniform}\ \mathsf{VPSPACE}^0 = \mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$. ∎

Enfin, nous montrons une conséquence (peu vraisemblable) de l'hypothèse que les classes $\mathsf{Uniform}\ \mathsf{VPSPACE}^0$ et $\mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$ coïncident.

4-V – Proposition

$$\mathsf{Uniform}\ \mathsf{VPSPACE}^0 = \mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}} \Longrightarrow \mathsf{PSPACE} = \mathsf{P\text{-}uniform}\ \mathsf{NC}.$$

Preuve Par la proposition 4-U, l'hypothèse implique déjà $\mathsf{P} = \mathsf{PSPACE}$. Nous allons montrer que $\oplus\mathsf{P} \subseteq \mathsf{P\text{-}uniform}\ \mathsf{NC}$ si l'on suppose l'égalité $\mathsf{Uniform}\ \mathsf{VPSPACE}^0 = \mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$. Il suffit pour cela de montrer que le langage $\oplus$HamiltonPath, qui est $\oplus\mathsf{P}$-complet, est dans $\mathsf{P\text{-}uniform}\ \mathsf{NC}$. Le problème $\oplus$HamiltonPath consiste à décider s'il y a un nombre impair de chemins hamiltoniens dans un graphe, voir Papadimitriou [63, p. 448]. Pour un graphe donné par sa matrice d'adjacence $(a_{i,j})_{1\leqslant i,j\leqslant n}$ (où $a_{i,j} = 1$ si et seulement s'il y a une arête entre i et j), le nombre de chemins hamiltoniens est

$$\sum_{1\leqslant j<k\leqslant n} \sum_{\sigma\in S_{j,k}} \prod_{i=1}^{n-1} a_{i,\sigma(i)},$$

où $S_{j,k}$ est l'ensemble de tous les n-cycles $\sigma \in \mathscr{S}_n$ qui commencent en j et terminent en k.[†] Le polynôme

$$p_n(x_{1,1},\ldots,x_{1,n},x_{2,1},\ldots,x_{n,n}) = \sum_{j<k} \sum_{\sigma\in S_{j,k}} \prod_{i=1}^{n-1} x_{i,\sigma(i)}$$

retourne ainsi le nombre de chemins hamiltoniens lorsque les variables $x_{1,1},\ldots,x_{1,n}$, $x_{2,1},\ldots,x_{n,n}$ encodent la matrice d'adjacence d'un graphe G. On voit facilement que cette famille de polynômes (p_n) est dans $\mathsf{Uniform}\ \mathsf{VPSPACE}^0$, qu'elle a un degré polynomial et que son évaluation modulo 2 répond à la question "$G \in \oplus$HamiltonPath ?".

Par notre hypothèse, (p_n) est dans $\mathsf{Uniform}\ \mathsf{VP}^0_{\mathsf{nb}}$ de sorte qu'il existe une famille uniforme (C_n) de circuits arithmétiques de taille polynomiale qui calcule (p_n). Remarquons que, malgré la borne polynomiale sur le degré de (p_n), cette famille de polynômes n'est pas nécessairement dans VP^0 car le degré formel complet de C_n n'est pas nécessairement polynomial (en effet, des constantes de taille exponentielle peuvent être calculées par C_n). C'est pourquoi nous ne pouvons pas évaluer directement C_n par l'algorithme de Miller, Ramachandran et Kaltofen [61]. Nous allons

[†]La condition $j \neq k$ impose de compter uniquement les chemins et non les cycles, et la condition $j < k$ permet d'éviter de compter deux fois chaque chemin, ce qui rendrait le problème $\oplus$HamiltonPath trivial.

construire une autre famille (D_n) de circuits qui calculent une famille de polynômes $(q_n) \in \mathsf{VP}^0$ telle que sur toute entrée booléenne, p_n et q_n ont la même parité.

L'idée est la suivante : on peut calculer seulement le reste modulo 2 des constantes car nous ne nous intéressons qu'au résultat modulo 2. Le circuit D_n est alors construit à partir de C_n comme suit. Premièrement, remarquons que le degré de p_n est $n-1$. On calcule chaque composante homogène de p_n séparément : chaque porte α de C_n est scindée en $n-1$ portes $\alpha_1, \ldots, \alpha_{n-1}$, la porte α_i calculant la composante homogène de degré i de α. La composante homogène de degré 0 (les constantes) n'est pas calculée, seul son reste modulo 2 est pris en compte. En d'autres termes, on remplace une constante paire par la constante 0 et une constante impaire par 1. Le circuit reste uniforme car on peut calculer en temps polynomial les constante modulo 2. La dernière étape de D_n est de calculer la somme de toutes les composantes homogènes (qui sont en nombre polynomial) de la porte de sortie.

Calculer ces composantes homogènes tout en gardant un circuit de taille polynomiale est facile et bien connu : on ignore simplement les composantes homogènes de degré $> n-1$. Avec cette construction, il est clair que p_n et q_n coïncident modulo 2, que la construction est uniforme et que le degré formel complet de D_n est au plus $n-1$ car il n'y a plus de constante dans le circuit. Ainsi $(q_n) \in \mathsf{VP}^0$.

Afin de décider le problème ⊕HamiltonPath, on doit donc seulement calculer la valeur de q_n modulo 2 sur l'entrée donnée, c'est-à-dire évaluer un circuit uniforme de taille polynomiale $s(n)$ et de degré formel complet majoré par $n-1$. Le théorème 5.3 de Miller, Ramachandran et Kaltofen [61] nous apprend qu'un tel circuit peut être évalué modulo 2 par un algorithme logspace-uniforme en temps parallèle $O(\log(s(n))\log(ns(n)))$, donc $O(\log(n)^2)$, et avec $O(n^2)$ processeurs, plaçant ainsi le problème ⊕HamiltonPath dans P-uniform NC^2.

Ainsi, en supposant que Uniform $\mathsf{VPSPACE}^0 =$ Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ nous avons montré que

$$\mathsf{PSPACE} = \mathsf{P} \subseteq \oplus\mathsf{P} \subseteq \mathsf{P}\text{-uniform } \mathsf{NC}^2,$$

ce qui termine la preuve.

On pourra remarquer au passage que la construction ne semble pas être logspace-uniforme car l'évaluation des constantes modulo 2 est un problème P-complet. Enfin, puisque nous construisons une famille de circuits qui est seulement P-uniforme, on aurait pu utiliser la construction de Valiant, Skyum, Berkowitz et Rackoff [82] plutôt que l'algorithme parallèle de [61]. En effet, comme il est précisé dans [61], la construction de [82] peut être réalisée en temps polynomial. ∎

4-W – Remarque Bien qu'extrêmement probable, la séparation des classes PSPACE et P-uniform NC ne semble pas être connue (pourtant, PSPACE peut être séparée de logspace-uniform NC par le théorème de hiérarchie en espace).

4.4.2 Bornes inférieures non-uniformes

Nous abordons maintenant la question « $\mathsf{VPSPACE} = \mathsf{VP}$? » du point de vue de la complexité booléenne. Le point de départ est la remarque suivante, similaire aux implications de la section précédente.

4-X – Proposition

$$\mathsf{VP}_{\mathsf{nb}} = \mathsf{VPSPACE} \Longrightarrow \mathsf{PSPACE} \subset \mathsf{NC/poly}.$$

Preuve La preuve de la proposition 4-V dans un contexte non-uniforme donne le résultat. ∎

Ainsi, si nous voulons montrer que $\mathsf{VPSPACE} \neq \mathsf{VP}_{\mathsf{nb}}$, nous pouvons essayer de montrer que $\mathsf{PSPACE} \not\subset \mathsf{NC/poly}$. Le théorème de hiérarchie en espace montre que c'est le cas dans un contexte uniforme, mais malheureusement il ne fonctionne pas pour les classes non-uniforme. De manière moins ambitieuse, on pourra plutôt essayer de montrer que $\mathsf{PSPACE} \not\subset \mathsf{L/poly}$. Il est facile de voir que cela est équivalent à $\mathsf{PSPACE/poly} \neq \mathsf{L/poly}$. Mais il s'avère que cette question est ouverte depuis un certain temps et qu'elle ne se relativise pas, ce qui souligne sa difficulté. Nous allons néanmoins montrer le résultat conditionnel suivant : sous une hypothèse issue de complexité de Kolmogorov en ressources bornées, ces deux classes sont différentes. Cette hypothèse s'appelle la symétrie de l'information. Pour la complexité de Kolmogorov usuelle (c'est-à-dire sans borne de temps ni d'espace), c'est un théorème de Levin [85] et Kolmogorov [49] qui affirme que pour tous mots x et y, x contient la même quantité d'information sur y que y sur x (à un terme logarithmique près). En revanche, lorsqu'on impose aux machines de fonctionner en temps polynomial ou en espace polylogarithmique, il s'agit d'une question ouverte.

Travaux antérieurs. Très peu de travaux concernent la question des classes en espace PSPACE et L/poly. En revanche, beaucoup concernent les classes en temps EXP et P/poly. Heureusement, les techniques utilisées jusqu'à présent pour traiter le cas des classes en temps fonctionnent aussi pour les classes en espace. Ainsi, ces deux questions semblent être très semblables et c'est pourquoi nous mentionnerons seulement des travaux concernant la complexité en temps. Notons que notre résultat est vrai également pour la complexité en temps, c'est d'ailleurs ainsi qu'il a été présenté dans [64]. Cependant, nous l'exposons ici dans le contexte de la complexité en espace pour coller à la définition de VPSPACE.

Mais voyons comment la question « $\mathsf{EXP} \subset \mathsf{P/poly}$? » a été abordée. Deux approches ont donné des résultats significatifs. Le but de la première est de trouver la plus petite classe non-uniforme qui ne contienne pas P/poly. Kannan [38] a montré que $\mathsf{NEXP}^{\mathsf{NP}}$ ne possède pas de circuits de taille polynomiale. Schöning [68] a donné par la suite une preuve plus simple du résultat plus faible que EXPSPACE n'a pas non plus de circuits de taille polynomiale : nous mentionnons ce résultat car nous reprendrons quelques idées de cette preuve. Cette première approche a tari assez vite car améliorer le résultat de Kannan implique d'obtenir une séparation qui ne se relativise pas ; le meilleur résultat connu est celui de Buhrman, Fortnow et Thierauf [12] : $\mathsf{MA}_{\mathsf{EXP}}$ et PEXP, les versions exponentielles de MA et PP respectivement, n'ont pas de circuits de taille polynomiale. La seconde approche consiste à trouver la meilleure borne inférieure non-uniforme pour EXP. Homer et Mocas [35] ont montré que EXP n'est pas inclus dans P/n^c pour toute constante c. Une preuve plus simple de ce dernier résultat est donnée dans [64].

Enfin, plus récemment, Lee et Romashchenko [50] ont obtenu un résultat du même esprit que celui de cette partie. En effet, ils prouvent que l'hypothèse de symétrie d'information en temps polynomial implique que $\mathsf{EXP} \neq \mathsf{BPP}$. La méthode que l'on présente

ici permet d'obtenir un résultat plus fort : $\mathsf{EXP} \not\subset \mathsf{P/poly}$. Cependant, l'hypothèse de symétrie de l'information n'est pas exactement la même dans les deux cas : dans [50], il s'agit de la forme « usuelle » où l'inégalité est plus serrée mais les bornes de temps plus généreuses.

Notons également que l'hypothèse de symétrie de l'information en temps polynomial avait déjà été reliée à des questions de complexité : notamment, Longpré et Watanabe [54] ont montré qu'elle était vraie si $\mathsf{P} = \mathsf{NP}$.

Plan de cette section. Nous commençons par quelques lignes définissant la complexité de Kolmogorov en ressources bornées. Puis nous introduisons et discutons l'hypothèse de symétrie de l'information et en voyons quelques propriétés, notamment son application itérée. Enfin, nous énonçons et montrons le résultat principal : la symétrie de l'information permet de séparer des classes non-uniformes.

Prérequis

Dans cette partie, nous présentons brièvement la complexité de Kolmogorov en ressources bornées et les classes à conseils. Pour plus de détails, on pourra se reporter au livre de Li et Vitányi [51] et à celui de Balcázar, Díaz et Gabarró [4].

Machine universelle On a tout d'abord besoin d'une machine universelle. Si M est une machine de Turing, on notera aussi M son codage booléen. Ainsi, M est vu tantôt comme une machine et tantôt comme un programme. Ce programme sera exécuté par une machine universelle comme suit.

4-Y – Proposition

> Il existe une machine de Turing universelle $\mathscr{U}$ à deux rubans qui, sur l'entrée (M, x), simule la machine M à deux rubans sur l'entrée x. De plus, il existe une constante $c > 0$, dépendant seulement de M, telle que si le calcul $M(x)$ nécessite s cellules sur M, alors la simulation de $M(x)$ prend cs cellules sur $\mathscr{U}$.

On fixe une telle machine universelle $\mathscr{U}$ pour le reste de ce chapitre. La machine M simulée par $\mathscr{U}$ sera aussi appelée le programme de $\mathscr{U}$. Par exemple, on dira que le programme M décide le langage A si pour tout x, le calcul $\mathscr{U}(M, x)$ s'arrête, et il accepte ssi $x \in A$.

On peut maintenant passer à la définition de la complexité de Kolmogorov. Si x et y sont deux mots et s est un entier, on note $C^s(x|y)$ la complexité de Kolmogorov en espace s de x sachant y, c'est-à-dire la taille du plus petit programme M qui, lorsqu'on le fait tourner sur la machine universelle $\mathscr{U}$, renvoie x sur l'entrée y en espace $\leqslant s$. En d'autres termes, le calcul $\mathscr{U}(M, y)$ prend moins de s cellules de travail et à la fin, le ruban de sortie contient le mot x. Pour une machine de Turing M (et en particulier pour $\mathscr{U}$), on notera $M^s(x)$ le mot écrit sur le ruban de sortie après le calcul de M sur x si celui-ci n'a pas utilisé plus de s cellules, de sorte qu'on peut écrire tout cela symboliquement :

$$C^s(x|y) = \min\{k \mid \exists M \text{ de taille } k \text{ tel que } \mathscr{U}^s(M, y) = x\}.$$

On notera $C^s(x)$ pour $C^s(x|\epsilon)$, où ϵ est le mot vide.

Classes à conseil Nous rappelons quelques notions du chapitre 1, mais du point de vue des machines universelles car c'est ce que nous utiliserons dans la suite. Si $s : \mathbb{N} \to \mathbb{N}$ est une fonction, la classe $\mathsf{DSPACE}(s(n))$ est l'ensemble des langages A reconnus en espace $O(s(n))$. Plus précisément, $A \in \mathsf{DSPACE}(s(n))$ s'il existe une constante $c > 0$ et un programme fixé $M \in \{0,1\}^*$ tels que pour tout mot x, le calcul $\mathscr{U}(M,x)$ utilise moins de $cs(|x|)$ cellules et il accepte ssi $x \in A$. On appelle PSPACE la classe $\cup_{k\geqslant 0}\mathsf{DSPACE}(n^k)$.

Maintenant, les calculs non-uniformes sont définis par des conseils, notion introduite par Karp et Lipton [39]. La classe à conseil $\mathsf{DSPACE}(s(n))/a(n)$ est l'ensemble des langages A tels qu'il existe un programme M, une constante $c > 0$ et une famille (a_n) de conseils (c'est-à-dire des mots) qui satisfont :

1. $|a_n| \leqslant a(n)$;
2. $\mathscr{U}(M,x,a_{|x|})$ fonctionne en espace $cs(|x|+a(|x|))$;
3. $\mathscr{U}(M,x,a_{|x|})$ accepte ssi $x \in A$.

La classe $\mathsf{L/poly}$ est définie par $\mathsf{L/poly} = \cup_{k\geqslant 0}\mathsf{DSPACE}(\log n)/n^k$ (c'est-à-dire espace de travail logarithmique et conseil de taille polynomiale). Similairement, $\mathsf{PSPACE/poly}$ est la classe $\cup_{k\geqslant 0}\mathsf{DSPACE}(n^k)/n^k$ (c'est-à-dire espace et conseil polynomiaux). On pourra remarquer que l'inclusion $\mathsf{PSPACE} \subset \mathsf{L/poly}$ est vraie si et seulement si $\mathsf{PSPACE/poly} = \mathsf{L/poly}$. Enfin, on notera que $\mathsf{NC/poly} \subseteq \cup_{k\geqslant 0}\mathsf{DSPACE}(\log^k n)/n^k$.

Conseils et programmes Pour une taille n fixée, les mots $x \in \{0,1\}^n$ de taille n sont ordonnés lexicographiquement et le i-ème est appelé $x^{(i)}$ (pour $1 \leqslant i \leqslant 2^n$). Soit A un langage. La *chaîne caractéristique* de $A^{=n}$ est le mot $\chi \in \{0,1\}^{2^n}$ défini par $\chi_i = 1$ ssi $x^{(i)} \in A$. Nous allons souvent considérer des programmes qui renvoient une chaîne caractéristique plutôt que de reconnaître un langage : cela reviendra essentiellement au même comme le montre le lemme suivant.

4-Z – Lemme

Si A est un langage de $\mathsf{DSPACE}(s(n))/a(n)$ (où $s(n) \geqslant \log n$), alors il existe des constantes $\alpha, k > 0$ et une famille (M_n) de programmes qui satisfont :

1. $|M_n| \leqslant k + a(n)$;
2. pour $1 \leqslant i \leqslant 2^n$ en binaire, $\mathscr{U}(M_n, i)$ renvoie les i premiers bits de la chaîne caractéristique χ de $A^{=n}$ en espace $\alpha(\log i + s(n + a(n)))$.

Preuve Soit M une machine $\mathsf{DSPACE}(s(n))$ qui décide A avec un conseil c de taille $a(n)$. La principale difficulté, s'il en est, est d'énumérer les i premiers mots de taille n sans les écrire (sinon on utiliserait un espace n). En fait, il suffit de ne pas écrire les zéros de début de mot : en effet, les i premiers mots ont tous $n - \log i$ zéros de tête. Ainsi, le programme M_n énumère simplement les i premiers mots x de taille n (sans les écrire entièrement) et simule $M(x,c)$: le programme M_n est donc composé du code de M, d'une routine d'énumération pour les i premiers mots de taille n et du conseil c de taille $a(n)$. ∎

Symétrie de l'information

Comme nous l'avons déjà dit, la symétrie de l'information a été montrée par Levin [85] et Kolmogorov [49] pour la complexité de Kolmogorov usuelle (ressources non bornées). En suivant la même preuve (voir par exemple Li et Vitányi [51, théorème 2.8.2 p. 182]), on montre aisément qu'elle est encore vraie lorsqu'on impose un espace de travail polynomial, comme l'affirme le théorème suivant.

4-AA – THÉORÈME

Il existe des constantes α et β telles que pour tous mots x, y et toute borne en espace s, l'égalité suivante est satisfaite :

$$C^s(x,y) \geqslant C^{\alpha s+(|x|+|y|)^\alpha}(x) + C^{\alpha s+(|x|+|y|)^\alpha}(y|x) - \beta \log(|x|+|y|).$$

Notons que l'autre inégalité est vraie également à un logarithme près et est bien plus facile à montrer.‡ Mais nous n'aurons besoin que de l'inégalité « difficile » par la suite. C'est une question ouverte de savoir si le théorème 4-AA est encore vrai pour une borne en espace polylogarithmique, c'est-à-dire s'il existe une constante α telle que

$$C^s(x,y) \geqslant C^{\alpha s+\log^\alpha(|x|+|y|)}(x) + C^{\alpha s+\log^\alpha(|x|+|y|)}(y|x) - \beta \log(|x|+|y|).$$

Cependant, s'il existe des fonctions à sens unique calculables en espace polylogarithmique, alors cette dernière inégalité est fausse (voir Longpré et Watanabe [54]). Ainsi, cette version polylogarithmique de la symétrie de l'information semble être trop restrictive.

On peut relâcher les contraintes de deux façons au moins : augmenter les bornes en espace d'une part, ou permettre un terme d'erreur plus élevé (c'est-à-dire plus que logarithmique) d'autre part. La première solution ne nous permet malheureusement plus de conclure car l'espace requis croîtrait trop rapidement. En revanche, la seconde solution permet d'avoir une hypothèse plus crédible et nous suffira. Dans un contexte de bornes sur le temps, on pourra voir la discussion à ce sujet dans Lee et Romashchenko [50] ou dans [64]. C'est ainsi que nous obtenons l'hypothèse suivante, baptisée (SI).

(SI) Il existe une constante $\alpha > 1/2$ et un polynôme q tels que pour toute borne s en espace et tous mots x, y, z de taille $|x|+|y|+|z| = n$,

$$C^s(x,y|z) \geqslant \alpha\Big(C^{s+q(\log n)}(x|z) + C^{s+q(\log n)}(y|x,z)\Big).$$

4-AB – Remarque On pourrait imposer d'avantage de restrictions sur l'hypothèse (SI) :

1. que $|x| = |y|$ et $C^{s+q(\log n)}(x|z) = C^{s+q(\log n)}(y|x,z)$;
2. que l'inégalité soit vraie seulement pour presque tous les mots x et y (c'est-à-dire pour tous sauf un nombre fini) ;
3. que α soit remplacée par le terme non-constant $1/2 + 1/\sqrt{\log(|x|+|y|)}$, plus proche de $1/2$.

‡La preuve de cette inégalité facile se généralise d'ailleurs aux bornes en espace polylogarithmiques.

En rassemblant ces trois points, on obtient une hypothèse plus faible. Si elle était fausse, la propriété suivante serait vérifiée : pour tout polynôme q il existe une borne s en espace, un nombre infini de mots x et y de même taille et un mot z tels que

$$C^{s+q(\log n)}(x|z) = C^{s+q(\log n)}(y|x,z) = k \text{ et pourtant } C^s(x,y|z) = k + o(k).$$

Enfin, notons que, contrairement au théorème 4-AA, la borne en espace du membre droit est maintenant $s + q(\log n)$ et non $O(s) + q(\log n)$ (il semble que la possibilité de multiplier s par une constante ne permette malheureusement plus de conclure) ; on pourrait aussi prendre $s + \log^{O(1)}(s) + q(\log n)$.

Voyons maintenant un lemme sur l'application itérée de (SI).

4-AC – Lemme

Supposons que (SI) soit vraie et prenons un polynôme q correspondant. Soit s une borne en espace, $u_1, \ldots, u_n$ des mots de taille t et z un autre mot de taille quelconque. On définit $m = nt + |z|$ la taille de tous ces mots. Supposons qu'il existe une constante k telle que pour tout $j \leqslant n$, on ait $C^{s+q(\log m)\log n}(u_j|u_1, \ldots, u_{j-1}, z) \geqslant k$.

Alors $C^s(u_1, \ldots, u_n|z) \geqslant (2\alpha)^{\lfloor \log n \rfloor} k$.

Preuve Prenons une suite de mots $(u_i)_{i \geqslant 1}$ de taille t. Montrons tout d'abord le résultat quand n est une puissance de 2. Nous montrons par récurrence sur n (et seulement pour les puissances de 2) la propriété suivante : pour toute borne en espace s, tout mot z et tout $m \geqslant nt + |z|$, si pour tout $j \leqslant n$, $C^{s+q(\log m)\log n}(u_j|u_1, \ldots, u_{j-1}, z) \geqslant k$, alors $C^s(u_1, \ldots, u_n|z) \geqslant (2\alpha)^{\log n} k$.

C'est vrai pour $n = 1$. Pour $n > 1$, fixons s, z et soit $m \geqslant nt + |z|$. Par (SI),

$$C^s(u_1, \ldots, u_n|z) \geqslant \alpha\Big(C^{s+q(\log m)}(u_1, \ldots, u_{n/2}|z) + C^{s+q(\log m)}(u_{n/2+1}, \ldots, u_n|u_1, \ldots, u_{n/2}, z) \Big).$$

Par hypothèse de récurrence au rang $n/2$, on sait minorer le second membre. En effet, il suffit de l'appliquer à la borne en espace $s + q(\log m)$ et de prendre comme nouveau z pour le second terme le mot $u_1, \ldots, u_{n/2}, z$. On obtient alors l'inégalité $C^s(u_1, \ldots, u_n|z) \geqslant \alpha((2\alpha)^{\log(n/2)} k + (2\alpha)^{\log(n/2)} k) = (2\alpha)^{\log n} k$, ce qui conclut lorsque n est une puissance de 2.

Maintenant, si n n'est pas une puissance de 2, soit p la plus grande puissance de 2 inférieure à n. Alors $C^s(u_1, \ldots, u_n|z) \geqslant C^s(u_1, \ldots, u_p|z) \geqslant (2\alpha)^{\log p} k = (2\alpha)^{\lfloor \log n \rfloor} k$.

∎

Nous établissons maintenant le lien entre la complexité de Kolmogorov de la chaîne caractéristique et la taille du conseil : le lemme simple suivant montre que si la première est grande alors on a besoin d'un grand conseil.

4-AD – Lemme

Soit A un langage et $\chi^{(n)}$ la chaîne caractéristique de $A^{=n}$ (c'est-à-dire $\chi^{(n)}_i = 1$ ssi $x^{(i)} \in A^{=n}$). On note $\chi^{(n)}[1..i]$ le mot formé des i premiers bits de $\chi^{(n)}$. Soit $r(n)$

une fonction et supposons qu'il existe une fonction non-bornée $s(n) \geqslant 0$ telle que pour toute constante $\alpha > 0$, il existe une infinité d'entiers n et il existe $1 \leqslant i \leqslant 2^n$ satisfaisant $C^{\alpha(\log i + r(n+a(n)))}(\chi^{(n)}[1..i]) > s(n) + a(n)$.
Alors $A \notin \mathsf{DSPACE}(r(n))/a(n)$.

Preuve Supposons que $A \in \mathsf{DSPACE}(r(n))/a(n)$. Il existe un programme fixé M et un conseil $c(n)$ de taille $\leqslant a(n)$ tels que pour tout $x \in \{0,1\}^n$, $\mathscr{U}(M, x, c(n))$ fonctionne en espace $O(r(n + a(n)))$ et accepte ssi $x \in A^{=n}$. En énumérant les i premiers mots de taille n dans l'ordre lexicographique (à nouveau, on n'écrit que les $\log i$ premiers bits de ces mots) et en simulant le programme M avec conseil $c(n)$ sur chacun d'eux, il existe un autre programme N avec conseil $c(n)$ qui énumère $\chi^{(n)}[1..i]$ en espace $O(\log i + r(n + a(n)))$. Ainsi il existe une constante $\alpha > 0$ telle que pour tout n et tout $i \leqslant 2^n$, $C^{\alpha(\log i + r(n+a(n)))}(\chi^{(n)}[1..i]) \leqslant |N| + a(n)$. ∎

Conséquence pour les classes non-uniformes

Nous pouvons maintenant montrer le résultat principal de cette partie : l'hypothèse de symétrie de l'information implique qu'il existe des langages décidables en espace polynomial mais pas en espace logarithmique, même avec un conseil de taille polynomiale. La principale difficulté est de diagonaliser sur tous les conseils de taille polynomiale sans pouvoir les énumérer (car on utiliserait alors un espace superpolynomial). L'idée est simple et peut être résumée comme suit.

- Deux parties « utiles et indépendantes » de taille k_1 et k_2 d'un conseil « doivent » contenir à elles deux environ $k_1 + k_2$ bits d'information.
- Ainsi, on peut décomposer un conseil en blocs de petite taille (par exemple $O(n)$), diagonaliser sur ces blocs puis les « recoller » grâce à (SI).

Comme on va le voir, cela revient à chercher efficacement une chaîne de grande complexité de Kolmogorov et le lemme 4-AC va nous y aider.

4-AE – THÉORÈME

Si (SI) est vraie alors $\mathsf{PSPACE} \not\subset \mathsf{NC/poly}$.

Preuve Supposons que (SI) soit vraie : cela nous donne un polynôme q. On va diagonaliser sur des blocs M (que l'on peut voir comme des programmes) de taille $\leqslant n-1$ et simuler la machine universelle $\mathscr{U}$ sur des calculs prenant un espace $s(n) = n$. On définit un langage A par taille de mots comme suit : pour le premier mot,

$$x^{(1)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié des blocs } M \text{ de taille } \leqslant n-1,\\ \text{le premier bit de } \mathscr{U}^{s(n)}(M) \text{ est } 0,\end{array}$$

c'est-à-dire qu'au moins la moitié des blocs M de taille $\leqslant n-1$ se trompent pour $x^{(1)}$. Soit $V_1^{(0)}$ l'ensemble des blocs M de taille $\leqslant n-1$ qui donnent la bonne réponse, c'est-à-dire tels que le premier bit de $\mathscr{U}^{s(n)}(M)$ corresponde à « $x^{(1)} \in A$ ». Ainsi, $|V_1^{(0)}| < 2^{n-1}$. On peut alors continuer avec $x^{(i)}$ pour $1 < i \leqslant n$, en divisant à chaque fois par deux le nombre de blocs qui ne se sont pas trompé jusqu'à présent :

$$x^{(i)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié des blocs } M \in V_{i-1}^{(0)},\\ \text{le } i\text{-ème bit de } \mathscr{U}^{s(n)}(M) \text{ est } 0,\end{array}$$

où $V_{i-1}^{(0)}$ est l'ensemble des blocs restants de taille $\leqslant n-1$ qui donnent la bonne réponse pour $x^{(i-1)}$. Remarquons que $V_n^{(0)} = \emptyset$ puisque tous les $2^n - 1$ blocs M ont été éliminés.

On appelle $u^{(1)}$ la chaîne caractéristique du segment initial de taille n de $A^{=n}$ défini par ce qui précède : ainsi, $|u^{(1)}| = n$ et pour tout $i \leqslant n$, $u_i^{(1)} = 1$ si et seulement si $x^{(i)} \in A^{=n}$. On définit maintenant le segment suivant de $A^{=n}$ de taille n.

$$x^{(n+1)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié de blocs } M \text{ de taille } \leqslant n-1,\\ \text{le premier bit de } \mathscr{U}^{s(n)}(M, u^{(1)}) \text{ est } 0,\end{array}$$

c'est-à-dire qu'au moins la moitié des blocs M de taille $\leqslant n-1$ se trompent pour $x^{(n+1)}$, même avec le mot $u^{(1)}$ comme conseil. Soit $V_1^{(1)}$ l'ensemble des blocs M qui ont raison pour $x^{(n+1)}$, i.e. tels que le premier bit de $\mathscr{U}^{s(n)}(M, u^{(1)})$ corresponde à « $x^{(n+1)} \in A$ ». Ainsi, $|V_1^{(1)}| < 2^{n-1}$. Et on continue comme ça :

$$x^{(n+i)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié des blocs } M \in V_{i-1}^{(1)},\\ \text{le } i\text{-ème bit de } \mathscr{U}^{s(n)}(M, u^{(1)}) \text{ est } 0.\end{array}$$

On appelle $u^{(2)}$ la chaîne caractéristique du deuxième segment de taille n de $A^{=n}$ défini ci-dessus : ainsi, $|u^{(2)}| = n$ et $u_i^{(2)} = 1$ si et seulement si $x^{(n+i)} \in A^{=n}$. Le troisième segment de taille n de $A^{=n}$ est alors défini de la même façon :

$$x^{(2n+1)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié des blocs } M \text{ de taille } \leqslant n-1,\\ \text{le premier bit de } \mathscr{U}^{s(n)}(M, u^{(1)}, u^{(2)}) \text{ est } 0,\end{array}$$

etc. En continuant ainsi, on obtient la définition suivante pour le $(j+1)$-ème segment de taille n de $A^{=n}$:

$$x^{(jn+i)} \in A^{=n} \iff \begin{array}{l}\text{pour au moins la moitié des blocs } M \in V_{i-1}^{(j)},\\ \text{le } i\text{-ème bit de } \mathscr{U}^{s(n)}(M, u^{(1)}, u^{(2)}, \ldots, u^{(j)}) \text{ est } 0.\end{array}$$

On arrête quand $j = n^{\log n}$ et décide arbitrairement que $x^{(k)} \notin A$ pour $k > n \times n^{\log n}$.

Montrons dans un premier temps que $A \notin \mathsf{DSPACE}(\log^k n)/\mathsf{poly}$ pour tout k. À chaque étape de la définition de A, on a $C^{s(n)}(u^{(j)} | u^{(1)}, \ldots, u^{(j-1)}) \geqslant n$ car aucun bloc de taille $\leqslant n-1$ n'écrit $u^{(j)}$ en espace $\leqslant s(n)$ sur l'entrée $u^{(1)}, \ldots, u^{(j-1)}$. Puisque nous supposons (SI) vraie et par définition de $s(n)$, le lemme 4-AC implique que pour $i = n \times n^{\log n}$, $C^{\log n(\log i + \log^k i)}(\chi^{(n)}[1..i]) \geqslant (2\alpha)^{\log^2 n} n = n^{1+\log(2\alpha)\log n}$ pour un n assez grand.

Ainsi, par le lemme 4-AD, en définissant $a(n) = n^{\log(2\alpha)\log n} - n$ et $r(n) = \log^k n$ on a $A \notin \mathsf{DSPACE}(r(n))/a(n)$. En particulier, $A \notin \mathsf{DSPACE}(\log^k n)/\mathsf{poly}$.

Enfin, on voit facilement que $A \in \mathsf{PSPACE}$ donc le théorème est montré. ∎

Pour revenir à la classe VPSPACE, on obtient donc le corollaire suivant en combinant le résultat précédent et la proposition 4-X.

4-AF – Corollaire

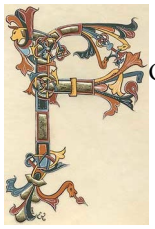
Si (SI) est vraie alors $\mathsf{VPSPACE} \neq \mathsf{VP}_{\mathsf{nb}}$.

Les résultats de ce chapitre suggèrent que la classe VPSPACE n'a pas de circuits de taille polynomiale. Malheureusement, nous n'avons pu le prouver, mais ce n'est pas étonnant au vu de la proposition 4-U, par exemple, qui relie cette question à des problèmes ouverts de complexité.

Et pourtant, cette question semble plus simple que dans le modèle BSS. En effet, dans les chapitres qui suivent, nous verrons que pour séparer le temps polynomial séquentiel du temps polynomial parallèle dans le modèle BSS, on doit d'abord séparer VP de VPSPACE. En d'autres termes, nous tisserons des liens entre VPSPACE et la classe PAR du modèle BSS grâce à deux théorèmes de transfert, l'un sur $\mathbb{C}$ et l'autre sur $\mathbb{R}$.

CHAPITRE
CINQ

Un théorème de transfert sur les complexes



ORTS des définitions du chapitre précédent, nous allons voir un deuxième résultat de transfert après celui du chapitre 3. Il concernera encore les classes de Valiant et de BSS, mais cette fois utilisera la classe VPSPACE. Il s'agit d'étudier les analogues de la question « $\mathsf{P} = \mathsf{PSPACE}$ » sur ces deux modèles de complexité algébrique : la version BSS de P est $\mathsf{P}_{\mathbb{C}}$ et de PSPACE est $\mathsf{PAR}_{\mathbb{C}}$ alors que pour Valiant c'est respectivement $\mathsf{VP}_{\mathsf{nb}}$ et VPSPACE. Ce résultat nous apprendra qu'il est au moins aussi difficile de séparer P de PSPACE dans le modèle BSS que dans celui de Valiant. Plus précisément, le théorème 5-L montre que $\mathsf{P}_{\mathbb{C}} \neq \mathsf{PAR}_{\mathbb{C}} \Rightarrow \mathsf{VP}_{\mathsf{nb}} \neq \mathsf{VPSPACE}$. Ainsi, le plus sage est sûrement de s'attaquer aux problèmes d'évaluation dans le cadre de Valiant.

La technique que l'on va utiliser est la même qu'au chapitre 3, c'est-à-dire qu'on va s'intéresser à la cellule de l'entrée plutôt qu'à l'entrée elle-même. Cependant, il s'agit ici de localiser l'entrée dans un arrangement d'hypersurfaces et non plus d'hyperplans, car la multiplication est maintenant prise en compte.

Travaux antérieurs. Comme résultats de transfert, nous pouvons citer ceux de Fournier et Koiran [27] et [28], où les théorèmes de transfert concernent PSPACE et la hiérarchie polynomiale en complexité booléenne et dans le modèle BSS. Ici toutefois, nous n'utiliserons pas de complexité booléenne mais plutôt le modèle de Valiant. La preuve du théorème de ce chapitre suit la construction de Koiran [42], où la condition de signe de l'entrée est trouvée par recherche dichotomique, mais la difficulté est de rendre la preuve constructive. On utilisera pour cela l'algorithme d'énumération des conditions de signe sur $\mathbb{C}$ de Fichtas, Galligo et Morgenstern [25], qui fonctionne en espace polynomial. De plus, la preuve de [42] utilise des nombres transcendants qu'il faudra remplacer par des entiers croissant suffisamment vite : il s'agit d'une technique classique héritée de Strassen [74] (voir aussi le livre de Bürgisser, Clausen et Shokrollahi [16]).

Plan du chapitre. Nous commençons par quelques résultats sur les conditions de signe sur $\mathbb{C}$, notamment l'algorithme d'énumération des conditions de signe satisfaisables de Fichtas, Galligo et Morgenstern [25]. Dans la partie 5.2, nous expliquons comment tester

l'appartenance d'un point à une variété grâce à des tests VPSPACE. Il s'agit de la partie la plus technique de ce chapitre. Puis la partie 5.3 consiste en l'énoncé et la preuve du théorème de transfert. Ce chapitre correspond à l'article [47].

5.1 Conditions de signe

5.1.1 Définition

Cela peut sembler étrange de parler de *signe* dans $\mathbb{C}$: en fait, nous entendons par-là la fonction qui vaut 0 ou 1 selon si le nombre complexe est respectivement nul ou non. La notion qui est au centre de nos techniques, que ce soit sur $\mathbb{C}$ ou sur $\mathbb{R}$ au chapitre suivant, est celle de *condition de signe*. Lorsque l'on a un ensemble $\{f_1, \ldots, f_s\}$ de s polynômes de $\mathbb{C}[x_1, \ldots, x_n]$, la condition de signe de $x \in \mathbb{C}^n$ est simplement le s-uplet constitué des signes que les s polynômes f_i prennent en x. La condition de signe S de x est donc le uple $S \in \{0,1\}^s$ défini par

$$S_i = 0 \iff f_i(x) = 0 \text{ pour } 1 \leqslant i \leqslant s.$$

Ainsi, une condition de signe est simplement un s-uplet de valeurs 0 ou 1. Cependant, elles ne correspondent pas toutes à des signes que peuvent réellement prendre les polynômes : on dit qu'une condition de signe est *satisfaisable* si c'est la condition de signe d'un élément $x \in \mathbb{C}^n$.

L'intérêt des conditions de signe deviendra plus clair à la partie 5.1.3. Pour l'instant, nous allons voir un résultat crucial : parmi toutes les 2^s conditions de signe, peu sont satisfaisables et de plus on peut les énumérer efficacement.

5.1.2 Énumération des conditions de signe

Grâce à un algorithme de tri parallèle et efficace (par exemple, l'algorithme de tri fusion de Cole [21]), on peut trier les conditions de signe satisfaisables en temps parallèle polylogarithmique. En tant que uples de 0 et de 1, les conditions de signe peuvent être vues comme des sous-ensembles de $\{1, \ldots, s\}$. On supposera que l'ordre choisi sur les conditions de signe est compatible avec l'inclusion, c'est-à-dire $S_1 \subseteq S_2 \Longrightarrow S_1 \leqslant S_2$. Par exemple, l'ordre lexicographique vérifie cette condition.

Le résultat fondamental ici provient de l'étude de l'élimination efficace des quantificateurs sur $\mathbb{C}$. Le théorème suivant découle de l'algorithme de Fichtas, Galligo et Morgenstern [25] : il y a « peu » de conditions de signe satisfaisables et on peut les énumérer « efficacement ». Notons que la borne sur le nombre de conditions de signes satisfaisables vient de Heintz [31].

5-A – THÉORÈME

Soient $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$ des polynômes et d leur degré maximal. Le nombre de conditions de signes satisfaisables est $N = (sd)^{O(n)}$ et il existe un algorithme fonctionnant en espace $(n \log(sd))^{O(1)}$ qui, sur l'entrée booléenne $f_1, \ldots, f_s$ (en représentation dense) et (i, j) en binaire, retourne la j-ème composante de la i-ème condition de signe satisfaisable.

Nous avons dit que ce théorème provenait de l'étude de l'élimination des quantificateurs sur $\mathbb{C}$. Nous aurons aussi besoin d'une version plus explicite à ce sujet, issue du même papier [25] : les quantificateurs peuvent être éliminés sans trop augmenter le degré et les coefficients des polynômes. Nous donnons seulement une version faible du résultat original : en particulier, nous n'avons pas besoin d'un algorithme mais seulement de bornes sur les coefficients et le degré.

On notera que dans notre contexte, les seules constantes utilisées dans nos formules sont 0 et 1, donc seuls des coefficients entiers peuvent apparaître. Pour utiliser des notations cohérentes avec la suite, on notera 2^s, 2^d et 2^{2^M} le nombre de polynômes, leur degré et la valeur absolue de leur coefficients, respectivement. Cette convention simplifiera les calculs et soulignera que s, d et M seront de polynômes. Précisons enfin que le polynôme $p(n,s,d)$ dans le théorème qui suit est indépendant de la formule ϕ.

5-B – THÉORÈME

Soit ϕ une formule du premier ordre, sur la structure $(\mathbb{C},0,1,+,-,\times,=)$, de la forme $\forall \bar{x}\,\psi(\bar{x})$, où $\bar{x}$ est un uple de n variables et ψ une formule sans quantificateur où 2^s polynômes apparaissent. Supposons que leurs degrés soient bornés par 2^d et leurs coefficients par 2^{2^M} en valeur absolue.

Il existe un polynôme $p(n,s,d)$, indépendant de ϕ, tel que la formule ϕ soit équivalente à une formule sans quantificateur Φ dans laquelle tous les polynômes ont un degré inférieur à $D(n,s,d) = 2^{p(n,s,d)}$ et dont les coefficients sont des entiers de valeur absolue majorée par $2^{2^M D(n,s,d)}$.

Dans notre étude des conditions de signe, nous manipulerons quelques variétés, ce qui nous amènera à des formules du premier ordre dont on voudra éliminer les quantificateurs, d'où ce théorème. Mais voyons pourquoi on s'intéresse aux conditions de signe.

5.1.3 Intérêt des conditions de signe

Le comportement d'une entrée $\bar{x} \in \mathbb{C}^n$ dans un circuit algébrique dépend uniquement des résultats de tous les tests du circuit. Ainsi, si l'on connaît l'ensemble des résultats de tous les tests, on peut savoir si l'entrée $\bar{x}$ est acceptée ou non. Or les tests effectués par une porte d'un circuit concernent la valeur de polynômes en $\bar{x}$. Si l'on connaît tous ces polynômes, il nous suffit alors de connaître la condition de signe de $\bar{x}$ pour savoir si $\bar{x}$ est accepté ou rejeté.

Ainsi, au lieu de manipuler l'entrée algébrique $\bar{x}$, on manipule simplement sa condition de signe, une entrée booléenne. Comme on le verra, cela nous permettra en quelque sorte d'éliminer les portes de tests de nos circuits algébriques et de les simuler par des circuits arithmétiques.

Afin d'adopter cette démarche, nous devons être capable d'énumérer tous les polynômes qu'un circuit peut tester. C'est l'objet de la proposition suivante, où l'on parcourt le circuit niveau par niveau comme dans Cucker et Grigoriev [23, théorème 3], utilisant les résultats des tests du niveau i pour calculer ceux du niveau $i+1$.

5-C – Proposition

Soit C un circuit algébrique sans constante, à n variables et de profondeur d.

1. Le nombre de polynômes différents pouvant être testés dans tous les calculs de C est $2^{d^2 O(n)}$.
2. Il existe un algorithme travaillant en espace $(nd)^{O(1)}$ qui, sur l'entrée C et les entiers (i,j) en binaire, calcule le j-ème bit du i-ème de ces polynômes.

Preuve On découpe C en niveaux correspondant à la profondeur des portes : les entrées sont au niveau 0 et la porte de sortie est la seule au niveau d.

Supposons que les résultats des tests des niveaux 0 à $i-1$ soient fixés : on peut alors calculer tous les polynômes testés au niveau i. Puisque les portes de nos circuits algébriques sont de degré entrant au plus 2, il y a au plus 2^{d-i} portes au niveau i de C : en particulier, au plus 2^{d-i} polynômes peuvent être testés au niveau i. Mais le degré d'un polynôme calculé au niveau i est au plus 2^i et la taille de ses coefficients est $(nd)^{O(1)}2^i$. Ainsi, par le théorème 5-A il y a au plus $(2^d)^{O(n)}$ résultats possibles pour les tests du niveau i et ils sont en outre énumérables en espace $(nd)^{O(1)}$.

On peut donc calculer tous les $(2^d)^{O(n)}$ résultats possibles de tous les tests du niveau i et continuer aux niveaux suivants. On obtient ainsi un algorithme travaillant en espace $(nd)^{O(1)}$ et qui énumère tous les polynômes qui peuvent être testés dans tous les calculs du circuit. Puisqu'il y a $2^{dO(n)}$ résultats possibles à chaque niveau, le nombre total de polynômes pour le circuit complet (pour d niveaux) est $(2^{dO(n)})^d = 2^{d^2 O(n)}$, ce qui prouve la proposition. ∎

On pourra remarquer que cette proposition peut aussi être utile pour des circuits avec constantes : il suffit de remplacer les constantes par de nouvelles variables. Le seul risque est en effet de prendre en compte plus de polynômes puisque nous avons remplacé des constantes spécifiques par des variables.

En combinant cette proposition avec le théorème 5-A, il est facile de montrer le résultat suivant, qui nous sera utile dans la preuve de la proposition 5-N : essentiellement, dans Uniform $\mathsf{VPSPACE}^0$ on peut énumérer les polynômes testés par le circuit ainsi que les conditions de signes satisfaisables correspondantes.

5-D – Lemme

Soit (C_n) une famille uniforme de circuits algébriques de profondeur polynomiale ayant un nombre polynomial d'entrées. On appelle $d(n)$ la profondeur de C_n et $i(n)$ le nombre d'entrées. Soient $f_1^{(n)},\dots,f_s^{(n)}$ tous les polynômes pouvant être testés par C_n comme dans la proposition 5-C, où $s = 2^{O(nd(n)^2)}$. Il y a donc $N = 2^{O(n^2 d(n)^2)}$ conditions de signes satisfaisables $S^{(1)},\dots,S^{(N)}$ par le théorème 5-A.

Alors il existe une famille $(g_n(\bar{x},\bar{y},\bar{z})) \in$ Uniform $\mathsf{VPSPACE}^0$, où $|\bar{x}| = i(n)$, $|\bar{y}| = O(n^2 d(n)^2)$ et $|\bar{z}| = O(nd(n)^2)$, telle que pour tout $1 \leqslant i \leqslant N$ et $1 \leqslant j \leqslant s$, on ait :

$$g_n(\bar{x},i,j) = \begin{cases} 0 & \text{si } S_j^{(i)} = 1 \\ f_j^{(n)}(\bar{x}) & \text{sinon.} \end{cases}$$

Preuve Calculer les coefficients des polynômes $f_1^{(n)},\dots,f_s^{(n)}$ prend un espace polyno-

mial par la proposition 5-C. En outre, décider si $S_j^{(i)} = 1$ se fait aussi en espace polynomial par le théorème 5-A. ∎

L'objectif maintenant est de trouver la condition de signe de l'entrée $\bar{x}$. Utiliser des tests Uniform VPSPACE^0 va nous permettre de le faire en temps polynomial ; mais pour cela, nous avons besoin de savoir grâce à ces tests si un point appartient à une variété. En effet, nous nous en servirons pour effectuer une recherche dichotomique afin de localiser la condition de signe de $\bar{x}$. C'est pourquoi la section qui suit s'attache au problème de décider si un point appartient à une variété.

5.2 Appartenance à une variété

Dans cette partie, nous expliquons comment effectuer dans Uniform VPSPACE^0 des tests d'appartenance de la forme « $\bar{x} \in V$ », où $V \subseteq \mathbb{C}^n$ est une variété. Nous commençons dans la section 5.2.1 par le cas où V est donnée par s polynômes. Dans ce cas, après quelques calculs préliminaires, on peut déterminer en $n+1$ tests si $\bar{x} \in V$.

Puis, dans la section 5.2.2, nous traitons le cas où V est donnée par l'union d'un nombre exponentiel de telles variétés : il s'agit du contexte de l'algorithme de la partie 5.3. Déterminer si $\bar{x} \in V$ requerra toujours $n+1$ tests, mais les calculs préliminaires serons plus lourds.

Nous aurons d'abord besoin des deux lemmes ci-dessous afin de réduire le nombre de polynômes et de remplacer des nombres transcendants par des entiers. Supposons qu'une variété V soit définie par les polynômes $f_1, \ldots, f_s$, où $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$. Quelle que soit la taille de s, nous devons décider si $\bar{x} \in V$ avec seulement $n+1$ tests. De manière non constructive, c'est possible et on pourra trouver une preuve dans Koiran [42] qui repose sur le lemme classique suivant : tout ensemble de $n+1$ combinaisons linéaires « génériques » des f_i définissent également V (ce résultat est vrai sur tout corps infini mais nous ne l'utiliserons ici que sur $\mathbb{C}$). Nous énonçons explicitement ce lemme car nous en aurons aussi besoin dans notre preuve constructive.

5-E – Lemme

Soient $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$ des polynômes et V la variété de $\mathbb{C}^n$ qu'ils définissent. Alors pour tous coefficients $(\alpha_{i,j})_{i=1..s, j=1..n+1} \in \mathbb{C}^{s(n+1)}$ algébriquement indépendants sur $\mathbb{Q}$, les $n+1$ combinaisons linéaires $g_j = \sum_{i=1}^{s} \alpha_{i,j} f_i$ (pour j variant de 1 à $n+1$) définissent V.

Malheureusement, dans notre cas nous ne pouvons pas utiliser des nombres transcendants (car nos circuits n'ont pas de constantes) et devons les remplacer par des entiers. Le lemme suivant de Koiran [41] nous assure qu'une suite d'entiers croissant suffisamment vite fera l'affaire. Nous énonçons ici une version plus faible adaptée à notre propos.

5-F – Lemme

Soit $\phi(\alpha_1, \ldots, \alpha_r)$ une formule du premier ordre sans quantificateur sur la structure $(\mathbb{C}, 0, 1, +, -, \times, =)$, contenant seulement des polynômes de degré $< D$ et dont les coefficients sont des entiers de valeur absolue $< C$. Supposons de plus que $\phi(\bar{\alpha})$ est vraie pour tous coefficients $(\alpha_1, \ldots, \alpha_r) \in \mathbb{C}^r$ algébriquement indépendants sur $\mathbb{Q}$.

> Alors $\phi(\bar{\beta})$ est vraie pour toute suite $(\beta_1, \ldots, \beta_r)$ d'entiers satisfaisant $\beta_1 \geqslant C$ et $\beta_{j+1} \geqslant CD^j \beta_j^D$ (pour $1 \leqslant j \leqslant r-1$).

La preuve peut être trouvée dans Koiran [41, lemme 5.4] et repose sur l'absence de grande racine réelle de polynômes à plusieurs variables.

Afin de donner une idée des preuves qui vont venir, essayons de montrer une version constructive du lemme 5-E, à savoir que $n+1$ polynômes à coefficients entiers et « efficacement calculables » suffisent à définir V. Cette première tentative ne fonctionnera pas mais permettra de comprendre l'intérêt des techniques qui suivront. L'idée est d'utiliser le lemme 5-F avec la formule $\phi(\bar{\alpha})$ affirmant que les $n+1$ combinaisons linéaires des f_i de coefficients $\alpha_{i,j}$ définissent la même variété que $f_1, \ldots, f_s$. Cette formule contient des quantificateurs, mais ils peuvent être éliminés sur $\mathbb{C}$ tout en gardant le degré et les coefficients raisonnablement petits grâce au théorème 5-B. Le lemme 5-E assure alors que $\phi(\bar{\alpha})$ est vraie dès que les $\alpha_{i,j}$ sont algébriquement indépendants. Le lemme 5-F nous dit donc que $\phi(\bar{\beta})$ est vraie pour tous entiers $\beta_{i,j}$ croissant suffisamment vite. Ainsi, V est maintenant définie par $n+1$ combinaisons linéaires à coefficients entiers des f_i, ce qui devrait conclure.

Malheureusement, cette tentative directe ne suffit pas car les coefficients en jeu deviennent trop grands pour être calculés en espace polynomial. C'est pourquoi nous devrons procéder par étapes dans les preuves ci-dessous : nous adopterons une approche « diviser pour régner » et des récurrences sur le nombre de polynômes.

5.2.1 Tests d'appartenance

Le cas de base de notre récurrence sera le lemme suivant, dont la preuve est esquissée à la fin de la partie précédente. Nous ne considérons en effet qu'un nombre restreint de polynômes ce qui évite le problème de coefficients et degrés trop élevés.

5-G – Lemme

> Il existe un polynôme $q(n,d)$ tel que, si $V \subseteq \mathbb{C}^n$ est une variété définie par $2(n+1)$ polynômes $f_1, \ldots, f_{2(n+1)} \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré $\leqslant 2^d$ et de coefficients bornés par 2^{2^M} en valeur absolue, alors :
>
> 1. la variété V est définie par $n+1$ polynômes $g_1, \ldots, g_{n+1} \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré $\leqslant 2^d$ et de coefficients bornés par $2^{2^{M+q(n,d)}}$ en valeur absolue ;
> 2. en outre, les bits des coefficients des g_i sont calculables en espace $Mq(n,d)$ à partir de ceux des f_j.

Preuve La formule du premier ordre $\phi(\bar{\alpha})$ (où $\bar{\alpha} \in \mathbb{C}^{2(n+1)^2}$), qui exprime que tout ensemble de $n+1$ combinaisons linéaires des f_j avec coefficients $\bar{\alpha}$ définit aussi V, peut s'écrire ainsi :

$$\phi(\bar{\alpha}) \equiv \forall x \in \mathbb{C}^n \left(\bigwedge_{i=1}^{n+1} \sum_{j=1}^{2(n+1)} \alpha_{i,j} f_j(x) = 0 \leftrightarrow \bigwedge_{j=1}^{2(n+1)} f_j(x) = 0 \right),$$

où $\alpha_{i,j}$ est l'abréviation de $\alpha_{2(i-1)(n+1)+j}$. Les polynômes de cette formule sont de degré $\leqslant 1+2^d$ et leurs coefficients sont bornés par 2^{2^M} en valeur absolue.

Sur $\mathbb{C}$, les quantificateurs de cette formule peuvent être éliminés par le théorème 5-B : $\phi(\bar{\alpha})$ est équivalente à une formule $\psi(\bar{\alpha})$ sans quantificateur, dans laquelle les polynômes ont un degré majoré par $D = D(n, \log(3(n+1)), d+1)$ et des coefficients majorés en valeur absolue par $C = 2^{2^M D}$, où $D(n, \log(3(n+1)), d+1) = 2^{p(n,\log(3(n+1)),d+1)}$ est défini dans le théorème 5-B.

Par le lemme 5-E, $\psi(\bar{\alpha})$ est vraie pour tous les coefficients $\bar{\alpha}$ algébriquement indépendants, et nous voudrions appliquer le lemme 5-F avec des entiers β_i croissant suffisamment vite. Soit $r = (1+2(n+1)^2)p(n, \log(3(n+1)), d+1)$, de sorte que

$$D \leqslant 2^r \text{ et } CD^{2(n+1)^2} \leqslant 2^{2^{M+r}}$$

et définissons

$$\beta_i = 2^{2^{M+2ir}} \text{ pour } 1 \leqslant i \leqslant 2(n+1)^2.$$

On remarquera que pour tout i, $\beta_i \leqslant 2^{2^{M+4(n+1)^2 r}}$ et il est facile de vérifier que $\beta_1 \geqslant C$ et $\beta_{i+1} \geqslant CD^i \beta_i^D$. Ainsi, par le lemme 5-F, $\psi(\bar{\beta})$ est vraie. On définit le polynôme $q(n,d) = 1 + 4(n+1)^2 r$ (à une constante multiplicative près pour la complexité en espace ci-dessous). Maintenant, poser

$$g_i = \sum_{j=1}^{2(n+1)} \beta_{i,j} f_j,$$

où $\beta_{i,j}$ est l'abréviation de $\beta_{2(i-1)(n+1)+j}$, montre le premier point du lemme.

Pour le second point, on remarquera que les bits des coefficients β_i sont calculables en espace $O(M + rn^2)$ et que les coefficients des g_i sont simplement une somme de $2(n+1)$ produits de β_j et de coefficients des f_k. Une telle multiplication nécessite seulement un espace $O(M + rn^2)$ puisque les entiers en jeu sont de taille $2^{O(M+rn^2)}$ (et même dans notre cas c'est particulièrement simple car les β_j sont des puissances de 2). Les $2n+1$ additions sont également effectuées en espace $O(M + rn^2)$, ce qui prouve le second point du lemme. ∎

Par récurrence, on peut maintenant étendre ce résultat à un plus grand nombre de polynômes.

5-H – Proposition

Il existe un polynôme $p(n,s,d)$ tel que, si V est une variété définie par 2^s polynômes $f_1, \ldots, f_{2^s} \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré $\leqslant 2^d$ et dont les coefficients sont bornés par 2^{2^M} en valeur absolue, alors :

1. la variété V est définie par $n+1$ polynômes $g_1, \ldots, g_{n+1} \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré $\leqslant 2^d$ et de coefficients majorés par $2^{2^{M+p(n,s,d)}}$ en valeur absolue ;
2. de plus, les bits des coefficients des g_i sont calculables en espace $Mp(n,s,d)$ à partir de ceux des f_j.

Preuve On raisonne par récurrence sur s. Soit $p(n,s,d) = sq(n,d)$ où $q(n,d)$ est le polynôme du lemme 5-G. Le cas de base $2^s \leqslant 2(n+1)$ découle du lemme 5-G. Supposons donc que $2^s > 2(n+1)$. On appelle V_1 et V_2 les variétés définies respectivement par $f_1,\ldots,f_{2^{s-1}}$ et $f_{2^{s-1}+1},\ldots,f_{2^s}$. Alors $V = V_1 \cap V_2$ et par hypothèse de récurrence, V_1 et V_2 sont toutes deux définies par $n+1$ polynômes de degré $\leqslant 2^d$ dont les coefficients sont majorés par $2^{2^{M+(s-1)q(n,d)}}$ en valeur absolue et sont calculables en espace $M(s-1)q(n,d)$.

Alors, par le lemme 5-G, V est définie par $n+1$ polynômes de degré $\leqslant 2^d$ dont les coefficients sont bornés par $2^{2^{M+sq(n,d)}}$ en valeur absolue et sont calculables en espace $Msq(n,d)$, comme annoncé. ∎

Cependant, les variétés que l'on manipulera seront données sous la forme d'une union d'un nombre exponentiel de variétés. C'est pourquoi nous devons travailler encore un peu pour obtenir des résultats fonctionnels. C'est l'objet de la partie suivante.

5.2.2 Union de variétés

Dans notre cas, les tests que l'algorithme devra effectuer dans la partie 5.3 ne sont pas exactement de la forme étudiée dans la section précédente : plutôt qu'une seule variété donnée par s polynômes, nous devrons décider si « $x \in W$? » lorsque $W \subseteq \mathbb{C}^n$ est l'union de k variétés. Bien sûr, puisque l'union est finie W est encore une variété, mais l'encodage n'est pas le même que précédemment : maintenant, il s'agit de k ensembles de s polynômes.

La première approche naïve est de définir $W = \cup_i V_i$ par les différents produits des polynômes qui définissent les V_i, mais il s'avère qu'il y a trop de tels produits. Nous allons plutôt adopter une méthode « diviser pour régner » avec une récurrence comme précédemment. Nous commençons par le cas le plus simple et qui nous servira d'étape d'induction : l'union de deux variétés de $\mathbb{C}^n$ peut encore être définie par $n+1$ polynômes.

5-I – Lemme

Il existe un polynôme $q(n,d)$ tel que, si V_1 et V_2 sont deux variétés de $\mathbb{C}^n$, chacune définie par $n+1$ polynômes de $\mathbb{Z}[x_1,\ldots,x_n]$, respectivement $f_1,\ldots,f_{n+1}$ et $g_1,\ldots,g_{n+1}$, de degré $\leqslant 2^d$ et dont les coefficients sont bornés par 2^{2^M} en valeur absolue, alors :

1. la variété $V = V_1 \cup V_2$ est définie par $n+1$ polynômes $h_1,\ldots,h_{n+1}$ appartenant à $\mathbb{Z}[x_1,\ldots,x_n]$, de degré $\leqslant 2^{d+1}$ et dont les coefficients sont bornés par $2^{2^{M+q(n,d)}}$ en valeur absolue ;
2. les bits des coefficients des h_i sont calculables en espace $Mq(n,d)$ à partir de ceux des f_j et g_k.

Preuve La variété V est définie par les $(n+1)^2$ polynômes $f_i g_j$ pour i et j compris entre 1 et $n+1$: ces polynômes ont un degré $\leqslant 2^{d+1}$. Remarquons par ailleurs qu'il y a au plus $2^{n(d+1)}$ monômes de degré fixé $\delta \leqslant 2^{d+1}$; ainsi, les coefficients des $f_i g_j$ sont des sommes d'au plus $2^{n(d+1)}$ produits d'entiers de taille 2^M. Il sont donc calculables en espace $O(Mnd)$ à partir de ceux des f_i et g_j. Cela montre aussi que les coefficients

des produits $f_i g_j$ sont majorés en valeur absolue par $2^{n(d+1)}2^{2^{M+1}} \leqslant 2^{2^{M+1+n(d+1)}}$. En appliquant la proposition 5-H, on conclut la preuve en posant $q(n,d) = 1 + n(d+1) + p(n, \log((n+1)^2), d+1)$, où p est le polynôme de la proposition 5-H. ∎

La généralisation à une union de plus de deux variétés suit par récurrence.

5-J – Proposition

Il existe un polynôme $r(n,s,k,d)$ tel que, si $V_1, \ldots, V_{2^k} \subseteq \mathbb{C}^n$ sont 2^k variétés, V_i étant définie par 2^s polynômes $f_1^{(i)}, \ldots, f_{2^s}^{(i)} \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré $\leqslant 2^d$ et de coefficients bornés par 2^{2^M} en valeur absolue, alors :

1. la variété $V = \cup_{i=1}^{2^k} V_i$ est définie par $n+1$ polynômes $g_1, \ldots, g_{n+1}$ appartenant à $\mathbb{Z}[x_1, \ldots, x_n]$, de degré $\leqslant 2^{d+k}$ et de coefficients bornés en valeur absolue par $2^{2^{M+r(n,s,k,d)}}$;
2. de plus, les bits des coefficients des g_i sont calculables en espace $Mr(n,s,k,d)$ à partir de ceux des $f_{j'}^{(j)}$.

Preuve Nous raisonnons par récurrence sur k. Soit $r(n,s,k,d) = (k+1)(p(n,s,d+k) + q(n,d+k))$, où p et q sont définis dans la proposition 5-H et lemme 5-I respectivement. Le cas de base $k = 0$ est une simple application de la proposition 5-H. Pour $k > 0$, nous appliquons d'abord la proposition 5-H aux V_i, de sorte que chaque variété V_i est maintenant définie par $n+1$ polynômes de degré $\leqslant 2^d$ et de coefficients majorés en valeur absolue par $2^{2^{M+p(n,s,d)}}$ et calculables en espace $Mp(n,s,d)$.
Groupons les variétés V_i par paires : pour $1 \leqslant i \leqslant 2^{k-1}$, on appelle $W_i = V_{2i-1} \cup V_{2i}$. Il y a 2^{k-1} variétés W_i et elles vérifient $V = \cup_i W_i$. Par le lemme 5-I, chaque variété W_i est définie par $n+1$ polynômes de degré $\leqslant 2^{d+1}$, de coefficients de taille $2^{M+p(n,s,d)+q(n,d)}$ et calculables en espace $M(p(n,s,d) + q(n,d))$. Par hypothèse de récurrence au rang $k-1$, V est définie par $n+1$ polynômes de degré $\leqslant 2^{d+1+(k-1)}$, de coefficients dont le log de la taille est $M + p(n,s,d) + q(n,d) + k(p(n, \lceil \log(n+1) \rceil, d+k-1) + q(n, d+k-1)) \leqslant M + r(n,s,k,d)$ et dont les bits sont calculables en espace $Mr(n,s,k,d)$. ∎

Nous pouvons enfin résumer tout cela par un résultat sur des tests VPSPACE permettant de décider l'appartenance à une union de variétés.

5-K – Corollaire

Soient $p(n)$ et $q(n)$ deux polynômes. Supposons que $(f_n(\bar{x}, \bar{y}, \bar{z}))$ soit une famille Uniform VPSPACE^0 où $|\bar{x}| = n$, $|\bar{y}| = p(n)$ et $|\bar{z}| = q(n)$. Pour tout entier $0 \leqslant i < 2^{p(n)}$, on appelle $V_i^{(n)} \subseteq \mathbb{C}^n$ la variété définie par les polynômes $f_n(\bar{x}, i, j)$ pour $0 \leqslant j < 2^{q(n)}$ (dans cette notation, i et j sont encodés en binaire).
Alors il existe une famille Uniform VPSPACE^0 $(g_n(\bar{x}, \bar{y}, \bar{z}))$, où $|\bar{x}| = n$, $|\bar{y}| = p(n)$ et $|\bar{z}| = \lceil \log(n+1) \rceil$, telle que

$$\forall \bar{x} \in \mathbb{C}^n, \ \forall k < 2^{p(n)}, \ \left(\bar{x} \in \bigcup_{i=0}^{k} V_i^{(n)} \iff \bigwedge_{j=0}^{n} g_n(\bar{x}, k, j) = 0 \right).$$

Preuve Si (f_n) est une famille Uniform VPSPACE^0, par définition il existe un polynôme $p(n)$ tel que le degré de f_n est majoré par $2^{p(n)}$ et la valeur absolue de ses coefficients par $2^{2^{p(n)}}$. Ainsi d, M, s et k sont polynomialement bornés dans la proposition 5-J et l'espace requis pour calculer les coefficients de g_n est polynomial. ∎

Ces différents résultats nous seront utiles dans la preuve du théorème de transfert ci-dessous.

5.3 Théorème de transfert

Nous voyons maintenant le principal théorème de ce chapitre, liant les questions de complexité dans le modèle BSS à celles dans le modèle de Valiant. Le théorème montre que séparer le temps polynomial parallèle du séquentiel dans le modèle BSS sur $\mathbb{C}$ implique de montrer que VPSPACE n'a pas de circuits de taille polynomiale. Bien que vraisemblablement très difficile, une telle séparation de classes de Valiant semble ainsi plus aisée que dans le modèle BSS. On peut également interpréter ce résultat comme une élimination des portes de test dans les circuits BSS sur $\mathbb{C}$.

5-L – THÉORÈME

> *Si* Uniform VPSPACE^0 = Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ *alors* $\mathsf{PAR}^0_{\mathbb{C}} = \mathsf{P}^0_{\mathbb{C}}$.

On remarquera que l'égalité des classes sans constante $\mathsf{PAR}^0_{\mathbb{C}}$ et $\mathsf{P}^0_{\mathbb{C}}$ implique $\mathsf{PAR}_{\mathbb{C}} = \mathsf{P}_{\mathbb{C}}$: il suffit de remplacer les constantes par de nouvelles variables de façon à transformer un problème $\mathsf{PAR}_{\mathbb{C}}$ en un problème $\mathsf{PAR}^0_{\mathbb{C}}$, puis de remplacer ces variables par leurs valeurs originales pour transformer un problème $\mathsf{P}^0_{\mathbb{C}}$ en un problème $\mathsf{P}_{\mathbb{C}}$.

5.3.1 Trouver la condition de signe de l'entrée

L'objectif est de trouver la condition de signe $S^{\bar{x}}$ de l'entrée $\bar{x} \in \mathbb{C}^n$. Nous allons donner un algorithme fonctionnant en temps polynomial, pouvant tester si la valeur d'un polynôme VPSPACE en $\bar{x}$ est nul. Tout d'abord, nous formalisons cette notion d'algorithme avec tests VPSPACE.

5-M – Définition

> Un algorithme polynomial avec tests Uniform VPSPACE^0 est la donnée d'une famille Uniform VPSPACE^0 $(f_n(x_1,\ldots,x_{u(n)}))$ et d'une famille uniforme (C_n) de circuits algébriques sans constante de taille polynomiale munis de portes de test spéciales de degré entrant $u(n)$, dont la valeur est 1 sur l'entrée $(a_1,\ldots,a_{u(n)})$ si $f_n(a_1,\ldots,a_{u(n)}) = 0$ et 0 sinon.

Remarquons qu'un nombre constant de familles Uniform VPSPACE^0 peuvent être utilisées dans cette définition (plutôt qu'une seule) : il suffit de les combiner toutes en utilisant des « variables de sélection ».

Voici maintenant le résultat précis que nous montrons : on peut trouver le rang de la condition de signe de $\bar{x}$ en temps polynomial grâce à des tests VPSPACE. Le « rang » d'une condition de signe satisfaisable est simplement son numéro dans l'ordre fixé sur les conditions de signe satisfaisables.

5-N – Proposition

Soit (C_n) une famille uniforme de circuits algébriques de profondeur polynomiale et ayant un nombre polynomial $i(n)$ d'entrées. Il existe un algorithme polynomial avec tests Uniform VPSPACE^0 qui, sur l'entrée $\bar{x} \in \mathbb{C}^{i(n)}$, retourne le rang i de la condition de signe $S^{(i)}$ de $\bar{x}$ par rapport aux polynômes $f_1, \ldots, f_s$ pouvant être testés par C_n, donnés par la proposition 5-C.

Preuve Soit $(g_n(\bar{x}, \bar{y}, \bar{z}))$ une famille Uniform VPSPACE^0 comme dans le lemme 5-D : essentiellement, g_n énumère à la fois tous les polynômes $f_1, \ldots, f_s$ pouvant être testés par C_n et les N conditions de signe satisfaisables $S^{(1)} < \ldots < S^{(N)}$. L'idée est de mener une recherche dichotomique pour trouver le rang i de la condition de signe de l'entrée $\bar{x}$.

Soit $S^{(j)} \in \{0,1\}^s$ une condition de signe satisfaisable. On dira que $S^{(j)}$ est une *candidate* si $\forall m \leqslant s$, $S^{(j)}_m = 0 \Rightarrow f_m(\bar{x}) = 0$. En particulier, la condition de signe de $\bar{x}$ est la plus petite candidate. Soit V_j la variété définie par les polynômes $\{f_m | S^{(j)}_m = 0\}$: par définition de g_n, V_j est aussi définie par les polynômes $g_n(\bar{x}, j, k)$ pour k variant de 1 à s. On remarquera que $S^{(j)}$ est une candidate si et seulement si $\bar{x} \in V_j$.

Le corollaire 5-K combiné avec le lemme 5-D montrent que les tests de la forme $\bar{x} \in \cup_{k \leqslant j} V_k$ sont dans Uniform VPSPACE^0. Ils sont alors utilisés pour effectuer une recherche dichotomique en faisant varier j. En un nombre d'étapes logarithmique en N (c'est-à-dire polynomial en n), on trouve le rang i de la condition de signe de $\bar{x}$. ∎

5.3.2 Un algorithme polynomial pour les problèmes $PAR_{\mathbb{C}}$

5-O – Lemme

Soit (C_n) une famille uniforme de circuits algébriques sans constante de profondeur polynomiale. Il existe un algorithme (booléen) travaillant en espace polynomial en n qui, sur l'entrée (i, n), décide si les éléments de la i-ème condition de signe satisfaisable $S^{(i)}$ sont acceptés par le circuit C_n.

Preuve On parcourt le circuit C_n niveau par niveau. Pour les portes de test, on calcule le polynôme f qui est testé. Puis on énumère les polynômes $f_1, \ldots, f_s$ comme dans la proposition 5-C pour le circuit C_n et on trouve le numéro j de f dans cette liste. En consultant le j-ème bit de la i-ème condition de signe satisfaisable par rapport à $f_1, \ldots, f_s$ (ce qui est fait par l'algorithme PSPACE du théorème 5-A), on connaît donc le résultat du test et on peut continuer ainsi jusqu'à la porte de sortie. ∎

5-P – THÉORÈME

Soit $A \in \mathrm{PAR}^0_{\mathbb{C}}$. Il existe un algorithme polynomial avec tests Uniform VPSPACE^0 *qui décide A.*

Preuve Le langage A est décidé par une famille uniforme (C_n) de circuits algébriques sans constante de profondeur polynomiale. Sur l'entrée $\bar{x}$, grâce à la proposition 5-N, on trouve d'abord le rang i de la condition de signe de $\bar{x}$ par rapport aux polynômes

$f_1, \ldots, f_s$ de la proposition 5-C. On conclut par un dernier test Uniform $\mathsf{VPSPACE}^0$ qui simule l'algorithme PSPACE du lemme 5-O sur l'entrée i. ∎

Le théorème de transfert 5-L est un corollaire immédiat de ce résultat.

Pour simplifier, on peut dire que l'on a un équivalent des classes P et PSPACE dans les modèles BSS et de Valiant. Les résultats de ce chapitre comparent les questions « $\mathsf{P} = \mathsf{PSPACE}\,?$ » sur $\mathbb{C}$ dans BSS et « $\mathsf{P} = \mathsf{PSPACE}\,?$ » dans Valiant. On soupçonne que la réponse à ces questions est négative, c'est-à-dire qu'on cherche à séparer P de PSPACE. Dans cette optique, le théorème 5-L montre que la séparation dans le modèle BSS est au moins aussi difficile que dans le modèle de Valiant.

Dans le prochain chapitre, nous obtenons un résultat similaire sur le corps $\mathbb{R}$ des réels.

CHAPITRE
SIX

Un théorème de transfert sur les réels

Ce chapitre est consacré à l'extension du résultat du chapitre précédent au corps $\mathbb{R}$ des réels. En d'autres termes, nous allons montrer que la séparation de P et PSPACE sur $\mathbb{R}$ dans le modèle BSS implique la séparation de P et PSPACE dans le modèle de Valiant. Plus précisément, nous montrons au théorème 6-J que $\mathsf{PAR}_{\mathbb{R}} \neq \mathsf{P}_{\mathbb{R}}$ implique $\mathsf{VPSPACE} \neq \mathsf{VP}_{\mathsf{nb}}$. Nous renvoyons à l'introduction du chapitre 5 pour de plus amples explications sur ce résultat.

Toute une partie technique de la preuve précédente sur $\mathbb{C}$ n'a plus lieu d'être. En effet, tester l'appartenance à une variété $V \in \mathbb{R}^n$ est très facile : il suffit de tester si la somme des carrés des polynômes est nulle. Ainsi, il semble plus simple d'établir le résultat sur $\mathbb{R}$. Cependant, il n'en est rien car il faut maintenant prendre en compte l'ordre sur $\mathbb{R}$, ce que nous n'avions pas à faire sur $\mathbb{C}$.

Les techniques seront toujours à base de conditions de signe, mais nous nous appuierons sur les résultats de Grigoriev [30] pour traiter le cas de l'ordre sur $\mathbb{R}$. Nous aurons pour cela besoin d'une version constructive du lemme de [30] montrant l'existence d'un vecteur orthogonal à environ la moitié d'une famille de vecteurs. La version constructive a été établie dans [19]. Ce lemme servira dans une recherche dichotomique à éliminer environ la moitié des vecteurs à chaque étape.

Plan du chapitre. Dans la première partie, nous présenterons la version constructive du lemme de Grigoriev [30]. Nous montrerons donc l'existence, pour toute famille de vecteurs sur le corps à deux éléments, d'un vecteur orthogonal à environ la moitié de la famille ; puis nous donnerons un algorithme parallèle efficace pour trouver un tel vecteur. La seconde partie sera consacrée au théorème de transfert proprement dit : nous utiliserons les conditions de signe et le vecteur orthogonal de la partie 6.1 pour prouver le résultat.

Ce chapitre s'appuie sur les articles [19] et [48].

6.1 Un vecteur orthogonal

Dans cette partie, nous améliorons les bornes du lemme de Grigoriev [30] : pour une famille de vecteurs de taille N, nous montrons qu'il existe un vecteur u orthogonal à au moins $N/2 - \sqrt{N}/2$ et au plus $N/2 + \sqrt{N}/2$ vecteurs de la famille. Le lemme original de Grigoriev donnait l'intervalle $[N/3; 2N/3]$. Et surtout, nous rendons le lemme constructif, c'est-à-dire que nous décrivons un algorithme efficace pour trouver un tel vecteur. Dans l'article [19], les bornes du lemme sont discutées, la preuve est dérandomisée et des algorithmes séquentiels simples sont décrits. Ici, nous nous contenterons de la preuve d'existence et d'un algorithme parallèle efficace.

6.1.1 Vecteurs et produit scalaire

On notera $\mathbb{F}_2$ le corps à deux éléments et nous allons considérer des vecteurs de ${\mathbb{F}_2}^d$, c'est-à-dire des d-uplets de zéros et uns. Pour $u \in {\mathbb{F}_2}^d$ et $1 \leqslant i \leqslant d$, on notera u_i le i-ème coefficient de u. On munit ${\mathbb{F}_2}^d$ d'un « produit scalaire » à valeur dans $\{0, 1\}$ défini de la façon suivante :

$$u \cdot v = \sum_{i=1}^{d} u_i v_i \bmod 2.$$

Bien sûr, on a mis des guillemets à « produit scalaire » puisqu'il s'agit seulement d'une forme bilinéaire, symétrique certes mais pas définie positive (par exemple, $u \cdot u = 0$ dès que le nombre de coefficients à 1 dans u est pair). Ce n'est donc pas un produit scalaire à proprement parler.

Soient $u, v \in {\mathbb{F}_2}^d$. Lorsque $u \cdot v = 0$, on dira que u et v sont orthogonaux. Si $V \subseteq {\mathbb{F}_2}^d$, on notera $V^\perp \subseteq {\mathbb{F}_2}^d$ l'ensemble des vecteurs w orthogonaux à tous les vecteurs de V.

6.1.2 Existence

Grâce à la méthode probabiliste, nous montrons ici, pour toute famille de vecteurs de ${\mathbb{F}_2}^d$, l'existence d'un vecteur $u \in {\mathbb{F}_2}^d$ orthogonal à environ la moitié des vecteurs de la famille. La preuve est non constructive ; un algorithme parallèle pour trouver u sera décrit par la suite. Comme on l'a déjà dit, l'intervalle donné ici améliore celui de Grigoriev [30] qui était $[N/3, 2N/3]$.

6-A – THÉORÈME

Soit $\mathscr{F}$ un ensemble fini de N vecteurs non nuls de ${\mathbb{F}_2}^d$. Il existe un vecteur $u \in {\mathbb{F}_2}^d$ tel que

$$-\frac{\sqrt{N}}{2} \leqslant |\{v \in \mathscr{F} \mid u \cdot v = 0\}| - \frac{N}{2} \leqslant \frac{\sqrt{N}}{2}. \tag{6.1}$$

Preuve Soient $v^{(1)}, \ldots, v^{(N)}$ les éléments de $\mathscr{F}$. On choisit un vecteur aléatoire $u \in {\mathbb{F}_2}^d$ en prenant, pour chacun des d coefficients, une probabilité $1/2$ qu'il soit à 0 et une probabilité $1/2$ qu'il soit à 1.

Soit Y_i la variable aléatoire définie par

$$Y_i = 1 \text{ si } u \cdot v^{(i)} = 0, \text{ et } Y_i = -1 \text{ sinon.}$$

Ainsi, nous nous intéressons à la variable aléatoire

$$Y = \sum_{i=1}^{N} Y_i = |\{i \mid u \cdot v^{(i)} = 0\}| - |\{i \mid u \cdot v^{(i)} = 1\}| = 2|\{i \mid u \cdot v^{(i)} = 0\}| - N.$$

Nous voulons montrer qu'il existe un vecteur u pour lequel $|Y| \leqslant \sqrt{N}$, c'est-à-dire $Y^2 \leqslant N$.

Montrons tout d'abord que $P(Y_i = 1) = 1/2$. Cela vient du fait que, pour tout vecteur $v^{(i)}$ fixé, il y a exactement le même nombre de vecteurs u tels que $u \cdot v^{(i)} = 0$ que de u tels que $u \cdot v^{(i)} = 1$: en effet, la dimension de $\{u \mid u \cdot v^{(i)} = 0\}$ est $d - 1$. Ainsi $E(Y_i) = 0$.

Montrons maintenant que les variables aléatoires Y_i sont *deux à deux*[§] indépendantes. Pour cela, considérons deux vecteurs $v^{(1)}$ et $v^{(2)}$ de $\mathscr{F}$. Nous devons montrer que pour tous $a, b \in \{-1, 1\}$,

$$P(Y_1 = a \cap Y_2 = b) = P(Y_1 = a)P(Y_2 = b) = 1/4. \tag{6.2}$$

Nous nous intéressons donc aux solutions $u \in {\mathbb{F}_2}^d$ du système

$$\begin{cases} v^{(1)} \cdot u &= a' \\ v^{(2)} \cdot u &= b' \end{cases}$$

où $a', b' \in \{0, 1\}$. Puisque $v^{(1)}$ et $v^{(2)}$ sont différents et non nuls, ils sont linéairement indépendants (nous sommes sur le corps à deux éléments) donc la dimension de l'espace des solutions du système affine précédent est $d - 2$ ce qui montre que la probabilité de l'intersection est $1/4$. On en déduit l'indépendance de Y_1 et Y_2.

Ainsi, puisque les variables aléatoires sont deux à deux indépendantes, on a $E(Y_i Y_j) = E(Y_i)E(Y_j) = 0$ si $i \neq j$. De plus, $E(Y_i^2) = 1$ donc par linéarité de l'espérance,

$$E(Y^2) = E(\sum_{i=1}^{N} Y_i^2 + \sum_{i \neq j} Y_i Y_j) = N.$$

Il existe donc un vecteur u pour lequel $Y^2 \leqslant N$. ∎

6-B – Définition

Un vecteur u vérifiant les inégalités du théorème sera appelé un *bon vecteur*.

6-C – Remarque L'indépendance deux à deux des Y_i nous permet d'évaluer la variance de Y : $\mathrm{Var}(Y) = \sum_{i=1}^{N} \mathrm{Var}(Y_i) = N$. Par l'inégalité de Tchebycheff, on a alors :

$$P(|Y - E(Y)| > 2\sqrt{N}) = P(|Y| > 2\sqrt{N}) < \mathrm{Var}(Y)/(2\sqrt{N})^2 = 1/4.$$

[§]On peut montrer qu'elles ne sont pas toujours trois à trois indépendantes.

Ainsi, au moins 3/4 des vecteurs u tombent dans l'intervalle $[N/2-\sqrt{N}, N/2+\sqrt{N}]$, ce qui fournit un algorithme probabiliste trivial pour trouver un tel vecteur. Toutefois, les méthodes habituelles pour déterminiser un tel algorithme parallèle semblent échouer, c'est pourquoi nous présentons ci-dessous une méthode *ad hoc*. De plus, notre algorithme déterministe donnera l'intervalle restreint $[N/2-\sqrt{N}/2, N/2+\sqrt{N}/2]$ du théorème.

6.1.3 Algorithme parallèle

Nous présentons maintenant un algorithme déterministe parallèle pour trouver un bon vecteur. En réalité, le but est de construire un algorithme fonctionnant en espace logarithmique ; nous utiliserons la correspondance temps parallèle / espace de travail mentionnée au chapitre 1 pour obtenir un algorithme utilisant un faible espace.

6-D – Remarque Nous savons par le résultat de Borodin, Cook et Pippenger [11] que l'espace probabiliste $f(n)$ est inclus dans l'espace déterministe $f(n)^2$. Puisque la remarque 6-C nous donne un algorithme probabiliste trivial et fonctionnant en espace logarithmique pour notre problème, le résultat de [11] semble nous donner un algorithme déterministe fonctionnant en espace polylogarithmique. Cependant, il ne semble pas possible d'utiliser directement ce résultat car il ne s'agit pas de problèmes de décision, et il n'est pas clair qu'on puisse l'adapter à notre cas. Nous devrons donc appliquer une technique *ad hoc* pour obtenir notre résultat.

L'idée est de construire *a priori* une petite famille de candidats, parmi lesquels on est sûr qu'un bon vecteur existe.

6-E – THÉORÈME

Il existe un algorithme parallèle qui, étant donnés deux entiers N et d vérifiant $N \leqslant 2^d$, construit en temps $O(\log N + \log d \log\log(dN))$ une famille $\mathscr{F}$ de $d^2N^2(N+1)^2$ éléments de ${\mathbb{F}_2}^d$ contenant, pour tous vecteurs distincts et non nuls $v^{(1)}, \ldots, v^{(N)} \in {\mathbb{F}_2}^d$, un vecteur u tel que

$$N/2 - \sqrt{N}/2 \leqslant |\{1 \leqslant i \leqslant N \mid v^{(i)}.u = 0\}| \leqslant N/2 + \sqrt{N}/2.$$

Une recherche exhaustive dans cette famille peut ainsi être effectuée en temps parallèle $O(\log(dN))$, ce qui nous permet de trouver un bon vecteur u sur l'entrée $v^{(1)}, \ldots, v^{(N)}$ en temps parallèle polylogarithmique $O(\log N + \log d \log\log(dN))$.

6-F – Remarque Allender nous indique que le facteur $\log\log(dN)$ peut être enlevé de la borne de complexité de notre algorithme, donnant ainsi un temps parallèle logarithmique. En effet, plutôt que d'utiliser une exponentiation rapide pour calculer θ^i dans la preuve ci-dessous, nous pouvons utiliser le résultat de Hesse, Allender et Barrington [34, corollaire 6.5] plaçant les calculs de division et de multiplication itérée de polynômes dans NC^1 (plus précisément dans Uniform TC^0 mais nous ne nous servirons pas de cette version forte ici) : cela nous donne un circuit de profondeur logarithmique pour calculer les θ^i.

Nous prouvons ce théorème dans la suite. Nous décrirons une famille logspace-uniforme de circuits de profondeur polylogarithmique pour résoudre notre problème. En entrée,

nous avons N vecteurs distincts et non nuls $v^{(1)},\dots,v^{(N)}$ de ${\mathbb{F}_2}^d$, donnés par leurs coordonnées. Ainsi, la taille de l'entrée est de l'ordre de Nd. La sortie sera un bon vecteur u pour $v^{(1)},\dots,v^{(N)}$. Comme indiqué dans le théorème, le principe de l'algorithme est de restreindre la recherche à un petit ensemble V dans lequel on est sûr qu'un bon vecteur u existe. Puisque cet espace de recherche est assez petit, on pourra alors trouver le bon vecteur par recherche exhaustive.

Nous montrons tout d'abord une condition à laquelle un ensemble contient un bon vecteur. Nous en aurons besoin pour construire notre famille de candidats.

6-G – Lemme

> Soit U un sous-espace orthogonal à aucun des $v^{(i)}-v^{(j)}$ (i.e. pour tous $1 \leqslant i < j \leqslant N$, il existe $v \in U$ tel que $v.(v^{(i)} - v^{(j)}) = 1$) et à aucun des $v^{(i)}$. Alors il existe un bon vecteur $u \in U$ pour $v^{(1)},\dots,v^{(N)}$.

Preuve Soit $u^{(1)},\dots,u^{(k)}$ une base de U. La condition que U n'est orthogonal à aucun des $v^{(i)} - v^{(j)}$ implique que les nouveaux vecteurs $w^{(i)}$ définis par $w^{(i)} = \sum_{l=1}^{k}(v^{(i)} \cdot u^{(l)})u^{(l)}$ sont tous distincts : en effet, pour tous $i \neq j$, il existe l tel que $v^{(i)} \cdot u^{(l)} \neq v^{(j)} \cdot u^{(l)}$. De plus, la condition que U n'est orthogonal à aucun des $v^{(i)}$ implique qu'aucun $w^{(i)}$ n'est nul. En termes géométriques, $w^{(i)}$ peut être vu comme la projection de $v^{(i)}$ sur U.

On définit maintenant sur U un nouveau produit $\odot : U \times U \to \mathbb{F}_2$ (associé à la base $(u^{(1)},\dots,u^{(k)})$) par la formule $(\sum_l \lambda_l u^{(l)}) \odot (\sum_l \mu_l u^{(l)}) = \sum_l \lambda_l \mu_l$. Pour ce nouveau produit (qui s'obtient par un simple changement de base par rapport au produit scalaire original), le théorème 6-A garanti l'existence d'un vecteur $u = \sum_l \lambda_l u^{(l)}$ qui est $\odot$-orthogonal à au moins $N/2 - \sqrt{N}/2$ vecteurs $w^{(i)}$ et au plus $N/2 + \sqrt{N}/2$. Mais on a : $u \odot w^{(i)} = \sum_l \lambda_l u^{(l)} \odot w^{(i)} = \sum_l \lambda_l (v^{(i)} \cdot u^{(l)}) = u \cdot v^{(i)}$, donc u est aussi un bon vecteur sur ${\mathbb{F}_2}^d$ (avec le produit scalaire usuel). ∎

Afin de construire un ensemble de candidats, nous allons travailler dans l'orthogonal. La condition du lemme précédent se traduit alors par des intersections plutôt que des produits scalaires ; d'où la notion de famille générique définie ci-après.

6-H – Définition

> Soit m un entier. Une famille $\mathscr{W} = \{W_1,\dots,W_k\}$ de sous-espaces de ${\mathbb{F}_2}^d$ est dite *générique* si pour tout ensemble $U \subseteq {\mathbb{F}_2}^d$ de m vecteurs non nuls, il existe un sous-espace $W_i \in \mathscr{W}$ tel que $W_i \cap U = \emptyset$.

Au début de l'algorithme, nous construisons une famille générique pour les ensembles de cardinal $N(N+1)/2$, dont les éléments $W_1,\dots,W_k$ sont des sous-espaces de grande dimension. Si l'on considère maintenant l'ensemble $U = \{v^{(i)} - v^{(j)} \mid 1 \leqslant i < j \leqslant N\} \cup \{v^{(i)} \mid 1 \leqslant i \leqslant N\}$, il est de cardinal $N(N+1)/2$, donc a une intersection vide avec un W_i particulier. Alors son orthogonal, $W_i^{\perp}$, vérifie les hypothèses du lemme 6-G, donc contient un bon vecteur. Si la dimension de W_i est suffisamment grande, son orthogonal sera petit donc on aura obtenu une petite famille de candidats dans laquelle on pourra effectuer une recherche exhaustive.

Nous voulons donc construire une famille générique. C'est l'objet du paragraphe suivant.

Famille générique

Nous montrons qu'une « bonne » famille générique $\mathscr{W} = \{W_1, \ldots, W_k\}$ pour des sous-ensembles de ${\mathbb{F}_2}^d$ de cardinal m existe et peut être construite efficacement en parallèle. Notre famille est de cardinal $k \leqslant 2dm$ et chaque sous-espace W_l est de dimension au moins $d - 1 - \log(dm)$. Les W_l seront donnés comme une intersection d'hyperplans, ce qui fournit immédiatement une famille génératrice de $W_l^\perp$.

Afin de disposer de plus de place, nous travaillerons d'abord dans une extension de $\mathbb{F}_2$. Plus précisément, nous fixons une extension K de degré $e = 1 + \lfloor \log((d-1)m) \rfloor$, de sorte qu'il y a plus de $(d-1)m$ éléments dans K. Nous identifierons K à $\mathbb{F}_2[X]/(P(X))$ pour un certain polynôme irréductible $P(X)$ de $\mathbb{F}_2[X]$ de degré e. Ainsi, les éléments de K seront vus comme des classes de polynômes modulo P. Une fois que le polynôme P est fixé, il est facile de calculer dans K, en manipulant des polynômes de degré $< e$ à coefficients dans $\mathbb{F}_2$ (on donnera les détails de l'implémentation par la suite).

Dans K^d, nous pouvons trouver $|K|$ hyperplans tels que tout sous-ensemble U de vecteurs non nuls de K^d de cardinal m a une intersection vide avec au moins l'un d'entre eux. Pour tout $\theta \in K$, considérons en effet l'hyperplan H_θ de K^d défini par l'équation $x_1 + \theta x_2 + \theta^2 x_3 + \cdots + \theta^{d-1} x_d = 0$. Il y a $|K| > (d-1)m$ hyperplans différents dans cette famille $(H_\theta)_{\theta \in K}$ et un point $a \in K^d \setminus \{0\}$ appartient à au plus $d-1$ hyperplans distincts : cela est dû au fait qu'il y a au plus $d-1$ racines distinctes au polynôme $P(\theta) = a_1 + a_2\theta + \cdots + a_d\theta^{d-1}$. Ainsi, parmi ces hyperplans, au moins un n'intersecte par U. Nous avons donc notre famille générique sur K.

Afin d'obtenir notre famille sur $\mathbb{F}_2$ (au lieu de K), nous considérons maintenant la « trace » de H_θ sur ${\mathbb{F}_2}^d$. Pour $(x_1, \ldots, x_d) \in {\mathbb{F}_2}^d$, l'équation de l'hyperplan H_θ peut être réécrite selon les puissances de X :

$$x_1 + \theta x_2 + \cdots + \theta^{d-1} x_d \equiv \sum_{i=0}^{e-1} \mu_i(x_1, \ldots, x_d) X^i \pmod{\mathrm{P}}$$

où les μ_i sont des combinaisons $\mathbb{F}_2$-linéaires des x_j (le coefficient de x_j dans μ_i est égal à la coordonnée X^i de θ^{j-1} dans la $\mathbb{F}_2$-base $1, X, \ldots, X^{e-1}$ de K). L'intersection $W_\theta = H_\theta \cap {\mathbb{F}_2}^d$ est alors définie par le système d'équations $\mu_i(x) = 0$ où i prend ses valeurs dans $\{0, 1, \ldots, e-1\}$. Ainsi, c'est un sous-espace de ${\mathbb{F}_2}^d$ de codimension au plus e.

Cette construction produit une famille $\mathscr{W} = \{W_1, \ldots, W_k\}$ de $k \leqslant 2^e$ sous-espaces de codimension $\leqslant e$ avec la propriété de généricité escomptée : pour tout sous-ensemble $U \subseteq {\mathbb{F}_2}^d \setminus \{0\}$ de cardinal m, il existe $W_l \in \mathscr{W}$ tel que $U \cap W_l = \emptyset$. Par le choix de e, on obtient au plus $2dm$ sous-espaces, de dimension au moins $d - 1 - \log(dm)$ chacun. De plus, ces sous-espaces sont donnés comme une intersection d'hyperplans.

Algorithme

Prenons maintenant $m = N(N+1)/2$ et considérons l'ensemble U de cardinal m défini comme suit :

$$U = \{v^{(i)} - v^{(j)} \mid 1 \leqslant i < j \leqslant N\} \cup \{v^{(i)} \mid 1 \leqslant i \leqslant N\}.$$

Pour ce choix de m, le paragraphe précédent nous fournit une famille générique $\mathscr{W} = \{W_1, \ldots, W_k\}$. En particulier, il existe un W_i tel que $U \cap W_i = \emptyset$. Alors $W_i^\perp$ vérifie les

hypothèses du lemme 6-G donc contient un bon vecteur pour $v^{(1)},\dots,v^{(N)}$. Or $W_i^\perp$ est de dimension au plus $e = 1+\lfloor\log((d-1)N(N+1)/2)\rfloor$ donc contient au plus $dN(N+1)$ éléments. Il suffit maintenant de faire une recherche exhaustive dans chacun des $W_j^\perp$ afin de trouver un bon vecteur. Voici donc les étapes de l'algorithme, son implémentation et analyse seront abordées au prochain paragraphe. L'entrée est un ensemble $\{v^{(1)},\dots,v^{(N)}\}$ de N vecteurs distincts et non nuls de ${\mathbb{F}_2}^d$, et la sortie est un vecteur u orthogonal à au moins $N/2-\sqrt{N}/2$ et au plus $N/2+\sqrt{N}/2$ d'entre eux.

1. Soit $e = \lfloor\log(dN(N+1))\rfloor$. En énumérant en parallèle tous les polynômes de $\mathbb{F}_2[X]$ de degré e, trouver un polynôme P irréductible. Soit $K = \mathbb{F}_2[X]/(P(X))$.
2. Soit $\mathscr{F}$ la famille d'hyperplans de K^d constituée des $|K| = 2^e$ hyperplans $(H_\theta)_{\theta\in K}$ décrits au paragraphe précédent. On réécrit l'équation de chaque hyperplan de $\mathscr{F}$ comme un système de e équations de $\mathbb{F}_2$. Il s'agit d'un simple réarrangement de termes. Nous obtenons un sous-espace W_θ de $(\mathbb{F}_2)^d$ de codimension au plus e pour chaque hyperplan H_θ. En tout, cette famille générique contient au plus 2^e sous-espaces de $(\mathbb{F}_2)^d$.
3. Effectuer en parallèle une recherche exhaustive dans $W^\perp$, pour tout W de la famille générique. Un bon vecteur doit exister dans au moins l'un d'entre eux (remarquons que seule cette dernière étape dépend de l'entrée).

Comme nous l'expliquons dans le paragraphe suivant, le temps d'exécution de cet algorithme est polylogarithmique en la taille dN de l'entrée.

Implémentation

Nous expliquons maintenant comment effectuer cette procédure rapidement en parallèle. Tout d'abord, afin de trouver un polynôme irréductible $P \in \mathbb{F}_2[X]$ de degré e, on énumère simplement tous les polynômes $A \in \mathbb{F}_2[X]$ de degré e et on teste leur irréductibilité. Il y a $2^e \leqslant dN(N+1)$ tels polynômes. Le polynôme A est irréductible si et seulement s'il n'est divisible par aucun autre polynôme non constant de degré $\leqslant e/2$. Cela donne un test d'irréductibilité immédiat : on calcule en parallèle les divisions euclidiennes de A par tout autre polynôme B non constant et de degré $\leqslant e/2$ et on teste si l'un des restes est nul. Trouver P se fait donc en temps parallèle $O(e)+T(e)$, où $T(e)$ est le coût d'une division dans $\mathbb{F}_2[X]$. Ainsi, nous avons besoin d'un algorithme de division en temps parallèle $O(e)$. Cette borne généreuse nous permet de tester en parallèle tous les quotients possibles Q et de vérifier si $A = BQ$. Il existe bien sûr des algorithmes de division bien plus rapides (mais trop compliqués pour notre propos), par exemple celui d'Eberly [24]. On pourrait aussi utiliser une version parallèle de l'algorithme de Berlekamp [7] pour trouver directement un polynôme irréductible.

Passons maintenant à la deuxième étape de l'algorithme, que l'on commence par un calcul préliminaire. Soit P le polynôme irréductible trouvé à la première étape et soit $K = \mathbb{F}_2[X]/(P(X))$ le corps à 2^e éléments. On calcule d'abord $X^i \bmod P$ pour tout $i \in [e, 2(e-1)]$. Le premier élément de cette suite est obtenu immédiatement à partir de P, et $X^{i+1} \bmod P$ est ensuite obtenu en temps constant à partir de $X^e \bmod P$ et $X^i \bmod P$ (il s'agit essentiellement d'un décalage de coefficients suivi d'au plus une addition dans $\mathbb{F}_2[X]$). La suite complète peut ainsi être construite en temps parallèle $O(e)$.

À l'étape 2, notre tâche principale est de calculer θ^i pour tout $i = 0, \ldots, d-1$ et tout $\theta \in K$. Par exponentiation rapide, θ^i peut être obtenu à partir de θ en $O(\log d)$ multiplications dans K, chacune de coût *booléen* $O(\log e)$. En effet, pour effectuer une telle multiplication, on multiplie deux polynômes de degré $\leqslant e-1$ à coefficients dans $\mathbb{F}_2$ et on prend le reste modulo $P(X)$. Le coût d'une multiplication dans $\mathbb{F}_2[X]$ est $O(\log e)$ et donne un polynôme de degré au plus $2(e-1)$. Au début de l'étape 2 nous avons précalculé une représentation modulo $P(X)$ de tous les monômes qui peuvent apparaître dans ce polynôme. Ainsi, il suffit d'ajouter au plus e polynômes de degré $\leqslant e-1$. Cela se fait en temps parallèle $O(\log e)$. Le coût total pour générer notre famille générique de 2^e sous-espaces est donc $O(\log d \log e)$, soit $O(\log d \log\log dN)$. L'orthogonal de chaque sous-espace W_θ contient au plus 2^e points puisque sa dimension est au plus e. Globalement, on a au plus $(2^e)^2$ points dans la réunion de tous les espaces orthogonaux. Puisque $2^e \leqslant dN(N+1)$, on obtient la borne $d^2N^2(N+1)^2$ du théorème 6-E. Le coût additionnel de l'énumération explicite de tous ces points est $O(\log e)$ puisque chaque point est la somme d'au plus e vecteurs générant un orthogonal.

Enfin, on peut trouver un bon vecteur parmi les $d^2N^2(N+1)^2$ candidats en temps $O(\log(dN))$ par recherche exhaustive. Dans un premier temps, on calcule en parallèle les produits scalaires $v^{(i)} \cdot u$ pour tous les vecteurs d'entrée $v^{(i)}$ et tous les candidats u. On peut faire cela en profondeur $O(\log d)$. Puis, pour un u fixé, on doit sommer sur tous les $v^{(j)}$ pour obtenir le nombre de i tels que $v^{(i)}.u = 1$. Un telle addition itérée peut être réalisée en profondeur $O(\log N)$ (voir par exemple le livre de Clote et Kranakis [20], en particulier la preuve du théorème 1.7.2). À cette somme, nous retranchons $N/2$ et prenons la valeur absolue, de sorte que pour tout candidat u nous avons calculé $\left|\left|\{1 \leqslant i \leqslant N \mid v^{(i)}.u = 1\}\right| - N/2\right|$. Il nous suffit maintenant de trouver le minimum parmi les $d^2N^2(N+1)^2$ valeurs ; on peut faire cela en profondeur $O(\log(d^2N^2(N+1)^2)) = O(\log(dN))$ puisque calculer le minimum est un problème AC^0 (on pourra voir l'exemple 6.2.2 de [20]). Ainsi, la recherche exhaustive requiert un temps parallèle $O(\log(dN))$ comme indiqué dans le théorème 6-E.

Le temps d'exécution total de notre algorithme est donc $O(\log N + \log d \log\log(dN))$ (en parallèle), ce qui prouve le théorème.

Espace logarithmique

Nous montrons ici que notre algorithme peut être exécuté sur un machine de Turing de sorte à utiliser un espace logarithmique seulement. Les trois étapes de l'algorithme peuvent en effet être réalisées en espace logarithmique.

- L'étape 1 consiste d'abord en une énumération des polynômes de degré e (donc logarithmique) et à coefficients dans $\mathbb{F}_2$. Cela prend un espace de travail $O(e)$. Puis il y a une autre énumération de polynômes avec un test de divisibilité. L'espace de travail est à nouveau $O(e)$.
- L'étape 2 consiste en des opérations arithmétiques dans K afin de calculer θ^i pour $0 \leqslant i \leqslant d$. Cela revient à des multiplications de polynômes de degré e et des réductions modulo $P(X)$: un espace logarithmique est encore suffisant.
- L'étape 3 calcule l'orthogonal d'un sous-espace W de codimension e donné par ses équations $(w_i.x = 0)_{i \leqslant e}$. Cet orthogonal $W^\perp$ est simplement l'espace engendré par les w_i. Énumérer les e coefficients pour ces vecteurs suffit donc à énumérer

tous les vecteurs de $W^{\perp}$. Tout cela se fait en espace $O(e)$. Enfin, vérifier si un vecteur est un bon vecteur peut encore être fait en espace logarithmique.

On a donc prouvé le résultat suivant.

6-I – THÉORÈME

Il existe un algorithme travaillant en espace $O(\log(dN))$ *qui, lorsqu'on lui donne une famille de* N *vecteurs de* ${\mathbb{F}_2}^d$*, retourne un bon vecteur* u *pour cette famille.*

6.2 Théorème de transfert

Nous pouvons maintenant étendre le théorème de transfert 5-L sur $\mathbb{C}$ du chapitre précédent au corps $\mathbb{R}$ des réels. De la même façon qu'au chapitre 5, si l'on parvient à séparer PAR de P sur $\mathbb{R}$, alors on aura séparé VP de VPSPACE.

6-J – THÉORÈME

Si $\mathsf{Uniform\ VP}^0_{\mathsf{nb}} = \mathsf{Uniform\ VPSPACE}^0$ *alors* $\mathsf{P}^0_{\mathbb{R}} = \mathsf{PAR}^0_{\mathbb{R}}$.

De même que sur $\mathbb{C}$, l'égalité des classes sans constante $\mathsf{PAR}^0_{\mathbb{R}} = \mathsf{P}^0_{\mathbb{R}}$ implique l'égalité $\mathsf{PAR}_{\mathbb{R}} = \mathsf{P}_{\mathbb{R}}$: il suffit de remplacer les constantes par de nouvelles variables pour transformer un problème $\mathsf{PAR}_{\mathbb{R}}$ en un problème $\mathsf{PAR}^0_{\mathbb{R}}$, puis de remplacer ces variables par leur valeur originale afin de transformer un problème $\mathsf{P}^0_{\mathbb{R}}$ en un problème $\mathsf{P}_{\mathbb{R}}$.

La preuve du théorème est une version constructive de Grigoriev [30] : elle s'appuie encore sur les conditions de signe sur $\mathbb{R}$ et aussi sur le vecteur orthogonal de la section précédente.

6.2.1 Conditions de signe

Comme au chapitre précédent, nous avons la notion de condition de signe, mais cette fois on ne distingue pas seulement parmi les deux cas $= 0$ et $\neq 0$, mais parmi les trois cas < 0, $= 0$ et > 0.

Ainsi, si on se donne s polynômes $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$, une condition de signe est un s-uplet $S \in \{-1, 0, 1\}^s$. Intuitivement, la i-ème coordonnée de S représente le signe de f_i : -1 pour < 0, 0 pour 0 et 1 pour > 0. La condition de signe d'un point $\bar{x} \in \mathbb{R}^n$ est donc le uple $S \in \{-1, 0, 1\}^s$ tel que $S_i = -1$ si $f_i(\bar{x}) < 0$, $S_i = 0$ si $f_i(\bar{x}) = 0$ et $S_i = 1$ si $f_i(\bar{x}) > 0$.

Comme précédemment, toutes les conditions de signe ne correspondent pas à un point $\bar{x} \in \mathbb{R}^n$ (par exemple, le polynôme $x^2 + 1$ n'est jamais négatif sur $\mathbb{R}$, donc la coordonnée correspondante ne sera jamais -1). On dit qu'une condition de signe est *satisfaisable* si c'est la condition de signe d'un point $\bar{x} \in \mathbb{R}^n$ et on note N le nombre de conditions de signe satisfaisables. Comme précédemment, le résultat clé, détaillé dans le paragraphe suivant, est un algorithme fonctionnant en espace polynomial pour énumérer les conditions de signe satisfaisables.

6.2.2 Un algorithme PSPACE pour énumérer les conditions de signe

Le résultat sur $\mathbb{R}$ suivant remplace l'algorithme sur $\mathbb{C}$ de Fichtas, Galligo et Morgenstern [25] du chapitre 5. La borne sur le nombre de conditions de signe satisfaisables est une conséquence des bornes de Thom-Milnor [62] (voir Grigoriev [29, lemme 1]) ; l'algorithme d'énumération en espace polynomial est dû à Renegar [67, proposition 4.1].

6-K – THÉORÈME

Soient s polynômes $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_n]$ de degré maximal d et dont les coefficients sont de taille binaire $\leqslant L$. Alors :

1. *il y a $N = (sd)^{O(n)}$ conditions de signe satisfaisables ;*
2. *il y a un algorithme travaillant en espace $(\log L)[n \log(sd)]^{O(1)}$ qui, sur l'entrée $(f_1, \ldots, f_s)$ en représentation dense et (i, j) en binaire, retourne la j-ème coordonnée de la i-ème condition de signe satisfaisable.*

Si S est la i-ème condition de signe satisfaisable dans l'énumération produite par cet algorithme, on dira que le *rang* de S est i (le rang est donc simplement le numéro de la condition de signe dans l'énumération). On remarquera que, si $d = 2^{n^{O(1)}}$, $s = 2^{n^{O(1)}}$ et $L = 2^{n^{O(1)}}$ comme ce sera le cas, alors l'espace de travail de l'algorithme est polynomial en n.

6.2.3 Énumérer tous les polynômes testés

Exactement comme la proposition 5-C sur $\mathbb{C}$, sur les réels aussi nous pouvons énumérer tous les polynômes qui peuvent être testés par un circuit algébrique. Rappelons que ce résultat remonte à [23, théorème 3].

6-L – Proposition

Soit C un circuit algébrique sans constante avec n variables et de profondeur d.

1. Le nombre de polynômes différents qui peuvent être testés par le circuit C est $2^{d^2 O(n)}$.
2. Il existe un algorithme travaillant en espace $(nd)^{O(1)}$ qui, sur l'entrée C et les entiers (i, j) en binaire, retourne le j-ème bit du codage du i-ème de ces polynômes.

Preuve Il s'agit de la même preuve que pour la proposition 5-C, en remplaçant l'algorithme d'énumération des conditions de signes de Fichtas, Galligo et Morgenstern [25] sur $\mathbb{C}$ par celui de Renegar [67] sur $\mathbb{R}$ (théorème 6-K ci-dessus). ∎

Ainsi, nous pouvons énumérer tous les polynômes qui peuvent être testés par un circuit algébrique. Si deux points $\bar{x}, \bar{y} \in \mathbb{R}^n$ ont la même condition de signe par rapport à ces polynômes, alors ils seront soit rejetés tous les deux par le circuit, soit acceptés tous les deux. C'est pourquoi nous pouvons manipuler les conditions de signe plutôt que les entrées algébriques $\bar{x}$ ou $\bar{y}$.

6.2.4 Preuve du théorème de transfert

Nous passons maintenant à la preuve du théorème 6-J. Soit $A \in \mathsf{PAR}^0_{\mathbb{R}}$: il est décidé par une famille uniforme (C_n) de circuits algébriques sans constante de profondeur polynomiale. À partir de maintenant, nous fixons n et travaillons sur le circuit C_n. Nous devons trouver la condition de signe de l'entrée $\bar{x}$ par rapport aux polynômes $f_1, \ldots, f_s$ de la proposition 6-L, c'est-à-dire par rapport à tous les polynômes qui peuvent être testés par C_n. On notera N le nombre de conditions de signe satisfaisables par rapport à $f_1, \ldots, f_s$.

Afin de trouver la condition de signe de l'entrée, nous allons donner un algorithme en temps polynomial qui peut tester la valeur d'une famille VPSPACE. Comme sur $\mathbb{C}$ (définition 5-M), on définit une notion d'algorithme polynomial avec tests VPSPACE.

6-M – Définition

> Un algorithme polynomial avec tests Uniform $\mathsf{VPSPACE}^0$ est la donnée d'une famille Uniform $\mathsf{VPSPACE}^0$ $(f_n(x_1, \ldots, x_{u(n)}))$ et d'une famille uniforme (C_n) de circuits algébriques sans constante et de taille polynomiale munis de portes de test spéciales, de degré entrant $u(n)$ et dont la valeur sur l'entrée $(a_1, \ldots, a_{u(n)})$ est 1 si $f_n(a_1, \ldots, a_{u(n)}) \leqslant 0$ et 0 sinon.

Comme pour la définition 5-M, un nombre constant de familles Uniform $\mathsf{VPSPACE}^0$ (plutôt qu'une seule) peuvent en fait être utilisées dans la définition précédente : il suffit de les combiner en utilisant des « variables de sélection ». L'étape principale pour montrer le théorème de transfert 6-J consiste en la preuve du résultat suivant. Il est prouvé dans la suite par une succession de lemmes : on procède comme dans Grigoriev [30] mais de manière constructive.

6-N – THÉORÈME

> *Il existe un algorithme polynomial avec tests* Uniform $\mathsf{VPSPACE}^0$ *qui, sur l'entrée* $\bar{x}$*, calcule le rang de la condition de signe de* $\bar{x}$ *par rapport aux polynômes* $f_1, \ldots, f_s$*.*

Conditions de signe partielles

Nous utiliserons la notion suivante issue de [30] : une *condition de signe partielle* est un élément T de $\{0,1\}^s$. Contrairement aux conditions de signe complètes, on ne distingue que les deux cas $= 0$ et $\neq 0$. On défini de manière naturelle la condition de signe partielle T d'un point $\bar{x} \in \mathbb{R}^n$: $T_i = 0$ si et seulement si $f_i(\bar{x}) = 0$.

Bien sûr, il y a moins de conditions de signe satisfaisables partielles que complètes, et bien sûr il existe aussi un algorithme fonctionnant en espace polynomial pour les énumérer. De plus, les conditions de signe partielles peuvent être vues comme des sous-ensembles de $\{1, \ldots, s\}$ (en utilisant la convention $k \in T \iff T_k = 1$), ce qui nous autorise à parler d'inclusion de conditions de signe partielles.

Comme dans le chapitre précédent, on fixe un ordre $\leqslant_T$ compatible avec l'inclusion et facilement calculable en parallèle, par exemple l'ordre lexicographique. On appelle alors $T^{(i)}$ la i-ème condition de signe partielle.

6-O – Lemme

Il existe un algorithme travaillant en espace polynomial en n qui, sur l'entrée (i,j) en binaire et $(f_1,\dots,f_s)$ en représentation dense, retourne la j-ème coordonnée de $T^{(i)}$ (la condition de signe partielle satisfaisable numéro i pour l'ordre $\leqslant_T$).

Preuve Il suffit d'utiliser l'algorithme du théorème 6-K, suivi d'une procédure parallèle efficace pour le tri, par exemple l'algorithme tri fusion de Cole [21]. ∎

On remarquera que la condition de signe partielle de l'entrée $\bar{x}$ est la condition de signe partielle T maximale satisfaisant $\forall i, T_i = 1 \Rightarrow f_i(\bar{x}) \neq 0$. Ainsi, nous recherchons un maximum, ce qui sera fait par une recherche dichotomique.

6-P – Lemme

Il existe une famille Uniform VPSPACE^0 (g_n) de polynômes qui satisfait, pour une entrée $\bar{x}$ réelle et i booléenne,

$$g_n(\bar{x},i) = \prod_{j\leqslant i}\Big(\sum_{k\notin T^{(j)}} f_k(\bar{x})^2\Big).$$

Preuve Le lemme 6-O montre que décider si $k \notin T^{(j)}$ se fait dans PSPACE. On utilise alors deux fois le lemme 4-O (une fois pour la somme et une fois pour le produit).

∎

6-Q – Proposition

Il existe un algorithme polynomial avec tests Uniform VPSPACE^0 qui, sur l'entrée $\bar{x}$, retourne le rang m de sa condition de signe partielle $T^{(m)}$.

Preuve L'algorithme consiste simplement en une recherche dichotomique grâce aux polynômes du lemme 6-P : si la condition de signe partielle de l'entrée $\bar{x}$ est $T^{(m)}$, alors $\prod_{j\leqslant i}(\sum_{k\notin T^{(j)}} f_k(\bar{x})^2) = 0$ si et seulement si $m \leqslant i$. En faisant varier i, on trouve m en un nombre d'étape logarithmique en le nombre de conditions de signe partielles satisfaisables, c'est-à-dire en temps polynomial. ∎

Conditions de signe complètes

On dit qu'une condition de signe (complète) S est compatible avec la condition de signe partielle T si $\forall i, T_i = 0 \Leftrightarrow S_i = 0$ (c'est-à-dire qu'elles sont d'accord pour « $= 0$ » et « $\neq 0$ »). Soit N' le nombre de conditions de signe satisfaisables compatibles avec la condition de signe partielle de l'entrée $\bar{x} \in \mathbb{R}^n$. Bien entendu, $N' \leqslant N$. Le lemme suivant est immédiat après le lemme 6-O et le théorème 6-K.

6-R – Lemme

Il existe un algorithme fonctionnant en espace polynomial en n qui, sur l'entrée (i,j,k), retourne le j-ème bit de la i-ème condition de signe satisfaisable compatible avec $T^{(k)}$.

Puisque nous connaissons la condition de signe partielle de $\bar{x}$ après avoir exécuté l'algorithme de la proposition 6-Q, nous connaissons quels polynômes sont nuls en $\bar{x}$. Nous pouvons donc ignorer les zéros dans les conditions de signe complètes compatibles. Ainsi, nous ne nous intéressons plus qu'à des conditions de signes à valeur dans $\{-1, 1\}$. Dans ce qui suit, nous allons utiliser de l'arithmétique sur le corps à deux éléments, il sera donc plus simple de considérer que nos conditions de signe prennent leur valeur dans $\{0, 1\}$ plutôt que $\{-1, 1\}$: 0 pour > 0 et 1 pour < 0. Ainsi, les conditions de signe sont maintenant des éléments de $\{0, 1\}^{s'}$, avec $s' \leqslant s$, que l'on pourra également considérer comme des sous-ensembles de $\{1, \ldots, s'\}$. Afin d'utiliser les résultats de la première partie de ce chapitre, l'ensemble $\{0, 1\}^{s'}$ est muni du « produit scalaire » $u.v = \sum_i u_i v_i \pmod 2$.

Nous récapitulons ici (avec les notations en cours) le résultat de la première partie de ce chapitre.

6-S – Proposition

Soit V un ensemble de N' vecteurs de $\{0, 1\}^{s'}$.

1. Il existe un vecteur u orthogonal à au moins $N'/2 - \sqrt{N'}/2$ et au plus $N'/2 + \sqrt{N'}/2$ vecteurs de V.
2. Un tel vecteur u peut être construit sur l'entrée V par un algorithme travaillant en espace logarithmique.

Rappelons que notre objectif est de trouver la condition de signe de $\bar{x}$. Nous allons utiliser la proposition 6-S afin de diviser par deux à chaque étape le cardinal de l'espace de recherche. On se fonde sur l'observation suivante : si $u \in \{0, 1\}^{s'}$, la valeur du produit $\prod_{j \in u} f_j(\bar{x})$ est négative si le produit scalaire de u et de la condition de signe de $\bar{x}$ est 1, et positive sinon. L'idée est alors de choisir u judicieusement de sorte que le nombre de conditions de signe satisfaisables ayant le même produit scalaire avec u que la condition de signe de $\bar{x}$ soit divisé par deux à chaque étape. Ainsi, en un nombre logarithmique d'étapes, la condition de signe de $\bar{x}$ sera déterminée de manière unique. Cela donne l'algorithme suivant pour trouver la condition de signe de $\bar{x}$.

- Soit E l'ensemble de toutes les conditions de signe satisfaisables ;
- tant que E contient plus d'un élément, faire
  - . trouver par la proposition 6-S un vecteur u orthogonal à au moins $|E|/2 - \sqrt{|E|}/2$ et au plus $|E|/2 + \sqrt{|E|}/2$ éléments de E
  - . soit b le résultat du test « $\prod_{j \in u} f_j(\bar{x}) < 0$? »
  - . prendre comme nouveau E l'ensemble de toutes les conditions de signe de E qui ont b comme produit scalaire avec u
- énumérer toutes les conditions de signe satisfaisables et trouver celle qui produit exactement les même résultats b que dans la boucle : il s'agit de la condition de signe de $\bar{x}$.

Le nombre d'étapes est $O(\log N')$, qui est polynomial en n. La dernière étape de l'algorithme (retrouver le rang de la condition de signe de $\bar{x}$ à partir des résultats de la boucle) est détaillée à la fin de cette section.

Nous montrons maintenant comment effectuer cet algorithme en temps polynomial avec des tests Uniform VPSPACE^0. La principale difficulté est que, selon la définition 6-M, nous ne pouvons utiliser qu'une seule famille VPSPACE (ou en tout cas seulement un nombre constant), alors qu'on aimerait faire des tests adaptatifs. Nous stockerons donc les résultats intermédiaires des tests précédents dans des variables $\bar{c}$ (une « liste de choix ») du polynôme VPSPACE. La proposition 6-S montre que, en réutilisant l'espace, il existe un algorithme logspace qui, étant donné un ensemble V de N' vecteurs et une « liste de choix » $c \in \{0,1\}^l$ (avec $l = O(\log N')$), énumère $l+1$ vecteurs $u^{(1)}, \ldots, u^{(l+1)}$ qui satisfont la condition $(\star)$ suivante.

- Le vecteur $u^{(1)}$ est orthogonal à au moins $N'/2 - \sqrt{N'}/2$ et au plus $N'/2 + \sqrt{N'}/2$ vecteurs de V.
- Soit $V_i \subseteq V$ le sous-ensemble de tous les vecteurs $v \in V$ qui satisfont $\forall j \leqslant i, v.u^{(j)} = c_j$. Alors le vecteur $u^{(i+1)}$ est orthogonal à au moins $|V_i|/2 - \sqrt{|V_i|}/2$ et au plus $|V_i|/2 + \sqrt{|V_i|}/2$ vecteurs de V_i.

Puisque $|V_i|$ est approximativement divisé par deux à chaque étape, le nombre d'étapes est $O(\log N')$. En particulier, puisque s' et N' sont simplement exponentiel, le lemme suivant découle facilement de ce qui précède et du lemme 6-R.

6-T – Lemme

Il existe un algorithme travaillant en espace polynomial en n qui, sur l'entrée codée en binaire (i,j,k,c), retourne le j-ème bit de $u^{(i)} \in \{0,1\}^{N'}$, où les vecteurs $u^{(1)}, \ldots, u^{(l+1)}$ satisfont la condition $(\star)$ ci-dessus pour l'entrée constituée de :

- l'ensemble V des N' conditions de signe satisfaisables compatibles avec $T^{(k)}$
- et de la liste de choix $c \in \{0,1\}^l$.

6-U – Lemme

Il existe une famille $(h_n) \in$ Uniform VPSPACE^0 qui satisfait, pour un point réel $\bar{x}$ et un triplet de mots booléens (i,k,c),

$$h_n(\bar{x}, i, k, c) = \prod_{j \in u^{(i)}} f_j(\bar{x}),$$

où $u^{(1)}, \ldots, u^{(l+1)}$ sont définis comme dans le lemme 6-T (en particulier, ils dépendent de $T^{(k)}$).

Preuve Le lemme 6-T montre que décider si $j \in u^{(i)}$ se fait en espace polynomial. On calcule alors le produit par le lemme 4-O. ∎

Ainsi, par un test Uniform VPSPACE^0, on connaît le signe du polynôme $h_n(\bar{x}, i, k, c) = \prod_{j \in u^{(i)}} f_j(\bar{x})$. On l'a vu précédemment, cela nous donne la valeur du produit scalaire de $u^{(i)}$ et de la condition de signe de $\bar{x}$: en effet, h_n est négatif si et seulement si ce produit scalaire est 1. En commençant par $c = 0 \ldots 0$ (à l'étape 1), et pour une étape $i \geqslant 2$, en définissant $c_{i-1} = 1$ si et seulement si le test précédent était négatif, le nombre de conditions de signe qui ont les mêmes produits scalaires que celle de $\bar{x}$ est divisé par

deux (environ) à chaque étape. À la fin, nous avons donc une liste de choix c à laquelle seule la condition de signe de $\bar{x}$ correspond. On a donc montré le lemme suivant.

6-V – Lemme

> Il existe un algorithme polynomial avec tests Uniform $\mathsf{VPSPACE}^0$ qui, sur l'entrée $\bar{x}$, retourne la liste des choix c (défini comme ci-dessus) qui caractérise uniquement la condition de signe de $\bar{x}$, si toutefois nous connaissons le rang k de la condition de signe partielle $T^{(k)}$ de $\bar{x}$.

On peut maintenant retrouver le rang de la condition de signe de $\bar{x}$ à partir de cette information, comme expliqué dans le paragraphe suivant.

Rang de la condition de signe

6-W – Lemme

> Il existe un algorithme travaillant en espace polynomial en n qui, sur l'entrée $c \in \{0,1\}^l$ (une liste de choix) et k, retourne le rang d'une condition de signe satisfaisable et compatible avec $T^{(k)}$ qui correspond à la liste de choix c.

Preuve En espace polynomial on peut recalculer tous les vecteurs $u^{(i)}$ comme dans le lemme 6-T, puis on énumère toutes les conditions de signe satisfaisables (grâce au théorème 6-K) jusqu'à ce qu'on en trouve une correspondant à liste de choix c. ∎

La preuve du théorème 6-N découle alors de la proposition 6-Q et des lemmes 6-V et 6-W.

Algorithmes polynomiaux pour $\mathrm{PAR}_{\mathbb{R}}$

On rappelle que A désigne un langage de $\mathsf{PAR}^0_{\mathbb{R}}$ et que (C_n) est une famille uniforme de circuits algébriques de profondeur polynomiale qui décide A. Le lemme suivant est le pendant du lemme 5-O sur les complexes.

6-X – Lemme

> Il existe un algorithme (booléen) fonctionnant en espace polynomial en n qui, sur l'entrée i (le rang d'une condition de signe satisfaisable) et n, décide si les éléments de la i-ème condition de signe satisfaisable S sont acceptés par le circuit C_n.

Preuve On parcourt le circuit C_n niveau par niveau. Pour les portes de test, on calcule le polynôme f à tester. Puis on énumère les polynômes $f_1, \ldots, f_s$ comme dans la proposition 6-L pour le circuit C_n et on trouve l'indice j de f dans cette liste. En consultant le j-ème bit de la i-ème condition de signe satisfaisable par rapport à $f_1, \ldots, f_s$ (ce que l'on fait grâce à l'algorithme du théorème 6-K), on sait donc le résultat du test et on peut continuer ainsi jusqu'à la sortie du circuit. ∎

6-Y – THÉORÈME

> *Soit* $A \in \mathsf{PAR}^0_{\mathbb{R}}$. *Il existe un algorithme polynomial avec tests* Uniform $\mathsf{VPSPACE}^0$ *qui décide* A.

Preuve Le langage A est décidé par une famille uniforme (C_n) de circuits algébriques de profondeur polynomiale. Sur l'entrée $\bar{x}$, grâce au théorème 6-N on trouve d'abord le rang de la condition de signe de $\bar{x}$ par rapport aux polynômes $f_1, \ldots, f_s$ de la proposition 6-L. Puis on conclut par le lemme 6-X. ∎

Le théorème de transfert 6-J suit immédiatement de ce résultat.

Conclusion et perspectives

ORNONS-NOUS à résumer notre travail et donner quelques perspectives. Le chapitre 2 est dans l'esprit d'Allender, Bürgisser, Kjeldgaard-Pedersen et Miltersen [2] et étudie la complexité (booléenne) de deux problèmes sur les circuits arithmétiques : calculer le degré d'un polynôme, d'une part, et décider si le coefficient d'un monôme est nul, d'autre part. Le résultat le plus intéressant de ce chapitre est probablement la proposition 2-N qui caractérise la complexité de ces problèmes si l'on travaille en arithmétique modulaire, et son corollaire, la proposition 2-U, qui améliore la borne supérieure connue pour le calcul du degré en caractéristique positive.

Les chapitres suivants montrent différents théorèmes de transferts. Au chapitre 3, on obtient au corollaire 3-AF que la séparation de P et NP dans le modèle BSS grâce à un langage de NP sans multiplication (sur un corps K de caractéristique nulle) implique la séparation de VP et VNP dans le modèle de Valiant (ou plus précisément une version uniforme et sans constante). On passe pour cela par l'intermédiaire de produits de taille exponentielle, d'une part, et d'interpolation de Lagrange, d'autre part.

Ce résultat ignore la multiplication dans NP car les techniques utilisées ne permettent de manipuler que des hyperplans et non des hypersurfaces. En effet, les algorithmes de manipulation d'hypersurfaces fonctionnent en espace polynomial en non en temps polynomial. Cela motive l'introduction de la classe VPSPACE au chapitre 4, soit en termes de fonctions coefficient calculables en espace polynomial, soit en termes de circuits arithmétiques de profondeur polynomiale. Ces deux caractérisations, ajoutées à celles de Poizat en termes de circuits de taille polynomiale comportant des portes d'évaluation ou de sommation, montrent qu'il s'agit d'une classe robuste et naturelle. Vraisemblablement, les familles de VPSPACE n'admettent pas de circuits arithmétiques de taille polynomiale ; l'un des résultats supportant cette conjecture est la proposition 4-T qui indique qu'il faudrait en effet à la fois que P/poly = PSPACE/poly en complexité booléenne et que VP = VNP. Encore plus frappant, il faudrait dans un contexte uniforme que PSPACE = P-uniform NC (proposition 4-V), ce que l'on pourrait presque infirmer grâce à un théorème de hiérarchie en espace, s'il n'y avait le problème de l'uniformité polynomiale de NC. Enfin, on notera que sous l'hypothèse de symétrie de l'information en espace polylogarithmique, on parvient à montrer que VPSPACE n'admet pas de circuits de taille polynomiale (corollaire 4-AF).

Aux chapitres 5 et 6, nous obtenons deux autres théorèmes de transferts (théorèmes 5-L et 6-J). Ils concernent cette fois les classes P et PAR du modèle BSS et VP et VPSPACE du modèle de Valiant, le premier sur le corps $\mathbb{C}$ des complexes et le second sur le corps $\mathbb{R}$ des réels. On montre en effet que la séparation de P et de PAR implique celle de VP et de VPSPACE. Pour les preuves, il s'agit essentiellement de rendre effectifs les résultats de Koiran [42] et Grigoriev [30] respectivement, en utilisant des algorithmes parallèles efficaces d'énumération des conditions de signe satisfaisables.

Pour résumer, la morale de ces résultats est que le modèle de Valiant est un bon point de départ pour étudier les questions ouvertes de complexité algébrique, car le modèle est plus simple et les réponses ne sont pas plus difficiles à obtenir. Ainsi se termine notre comparaison des questions sur les problèmes de décision et d'évaluation en complexité algébrique, pour laisser la place à l'énoncé de quelques questions ouvertes et pistes de recherche future.

Perspectives

Directement liés à nos résultats, certaines questions sont très naturelles à la fin de ce manuscrit. Nous commençons par le chapitre 2 : le problème du calcul du degré d'un polynôme n'est vraisemblablement pas si difficile que le suggère la borne supérieure CH.

- ▹ Donner un meilleur algorithme ou une meilleure borne inférieure pour le calcul du degré d'un polynôme donné par un circuit arithmétique.

En ce qui concerne la classe VPSPACE, le passage à la complexité booléenne au chapitre 4, notamment la proposition 4-V, amène bien sûr le problème suivant.

- ▹ Montrer que $\mathsf{PSPACE} \neq \mathsf{P\text{-}uniform\ NC}$.

Cette situation est si proche du cas d'application du théorème de hiérarchie en espace qu'il semble en effet difficile de pencher pour l'égalité des deux classes…

Toujours concernant la complexité booléenne, la fin du chapitre 4 suit l'article [64] et traite des liens avec la symétrie de l'information. Dans [64], nous obtenons $\mathsf{EXP} \not\subset \mathsf{P/poly}$ sous l'hypothèse de symétrie de l'information en temps polynomial. Est-il possible d'utiliser des méthodes semblables pour montrer un résultat inconditionnel, c'est-à-dire trouver la plus petite classe possible n'ayant pas de circuits de taille polynomiale ? Le meilleur résultat connu dans cette direction est celui de Buhrman, Fortnow et Thierauf [12] : $\mathsf{MA_{EXP}}$ et PEXP, les versions exponentielles de MA et PP respectivement, n'ont pas de circuits de taille polynomiale. Peut-on améliorer ce résultat ?

- ▹ Montrer un résultat inconditionnel du genre $\mathsf{EXP}^{\mathsf{BPP}} \not\subset \mathsf{P/poly}$.

Mais ces questions de complexité booléenne semblent un peu moins proches de l'objet de ce manuscrit. Revenant à la complexité algébrique, nos théorèmes de transfert sont des implications et non des équivalences ; qu'en est-il de la réciproque ?

- ▹ La réciproque des théorèmes 5-L et 6-J est-elle vraie ?

Cette question semble plus facile sur le corps des complexes car l'emploi du Nullstellensatz semble plus aisé que celui du Positivstellensatz sur les réels. Toutefois, seules des réponses partielles sont connues. Sur $\mathbb{C}$, un argument de chemin canonique similaire à Shub et Smale [70] montre la réciproque à un multiple près, c'est-à-dire que, si

$\mathsf{PAR}^0_{\mathbb{C}} = \mathsf{P}^0_{\mathbb{C}}$, toute famille $(f_n) \in \mathsf{Uniform\ VPSPACE}^0$ a un multiple non nul $(g_n f_n)$ dans $\mathsf{Uniform\ VP}^0_{\mathsf{nb}}$. Sur les réels ou les complexes, Bürgisser [14] aborde cette question par la notion de *complexité d'approximation*, introduite par Strassen [74]. Il s'agit non pas de calculer un polynôme $f(\bar{x})$ exactement, mais plutôt de calculer une « approximation » $g(\bar{x}, y)$ de f telle que, en gros, $\lim_{\epsilon \to 0} g(\bar{x}, \epsilon) = f(\bar{x})$ (voir [14] pour plus de détails). Dans ce contexte, on obtient une extension naturelle de la classe VP, appelée $\underline{\mathsf{VP}}$, constituée des familles de polynômes de degré et de complexité approximative polynomiaux. Le corollaire 4.5 de Bürgisser [14] montre alors que

$$\mathsf{P}_{\mathbb{R}} = \mathsf{PAR}_{\mathbb{R}} \Rightarrow \mathsf{VNP} \subseteq \underline{\mathsf{VP}},$$

c'est-à-dire une réciproque faible du théorème 6-J (ce résultat est également vrai sur $\mathbb{C}$). Il serait intéressant de situer la classe $\underline{\mathsf{VP}}$ parmi les autres classes de Valiant connues.

Dans un cadre plus général et pour conclure, mentionnons deux questions qui n'ont pas été abordées dans ce manuscrit. Malgré le grand nombre de résultats de transfert connus concernant le modèle de Valiant, avec BSS d'une part, mais aussi avec les classes booléennes d'autre part, aucun transfert des booléens vers Valiant n'est connu. Il serait ainsi intéressant de trouver une hypothèse de nature combinatoire qui impliquerait l'égalité de deux classes de Valiant.

- ▹ Trouver une hypothèse de nature combinatoire impliquant $\mathsf{VP} = \mathsf{VNP}$.

Enfin, terminons par une question de Papadimitriou, étudiée dans [46] et qui mérite plus d'attention. Les circuits arithmétiques sont limités aux opérations d'addition et de multiplication, tandis qu'un algorithme booléen est capable de faire des opérations plus complexes, comme des divisions, des pgcd, etc. Si l'on sait évaluer un polynôme par un algorithme booléen, peut-on alors le calculer par un circuit arithmétique ? Une version plus précise de cette question est donnée ci-dessous. Éliminer la division des circuits arithmétiques s'est avéré possible : en utilisant le développement en série entière de $1/(1+x)$, Strassen [73] (voir aussi le livre de Bürgisser, Clausen et Shokrollahi [16, chapitre 7]) répond affirmativement à la question pourvu que le degré du polynôme soit polynomial. Peut-on en faire autant pour toutes les opérations calculables en temps polynomial ?

- ▹ Si, en temps polynomial, on sait évaluer aux points rationnels un polynôme à coefficients entiers, est-il vrai que ce polynôme admet un circuit arithmétique de taille polynomiale ?

Bibliographie

[1] L. M. Adleman : Two theorems on random polynomial time. *In Proc. 19th IEEE Symposium on Foundations of Computer Science*, p. 75–83, 1978. Cité à la page 43

[2] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen et P. B. Miltersen : On the complexity of numerical analysis. *In IEEE Conference on Computational Complexity*, p. 331–339, 2006. Cité à la page 25, 36, 37, 44, 107

[3] L. Babai et L. Fortnow : Arithmetization : A new method in structural complexity theory. *Computational Complexity*, 1:41–66, 1991. Cité à la page 61

[4] J. L. Balcázar, J. Díaz et J. Gabarró : *Structural Complexity I*. Num. 11 de EATCS monographs on theoretical computer science. Springer-Verlag, 1988. Cité à la page 7, 71

[5] R. Beigel et J. Gill : Counting classes : Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1):3–23, 1992. Cité à la page 9

[6] S. J. Berkowitz : On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984. Cité à la page 63

[7] E. R. Berlekamp : Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967. Cité à la page 97

[8] L. Blum, F. Cucker, M. Shub et S. Smale : *Complexity and Real Computation*. Springer-Verlag, 1998. Cité à la page 23, 40

[9] L. Blum, M. Shub et S. Smale : On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989. Cité à la page 23

[10] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo et M. Tompa : Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, 1989. Cité à la page 36

[11] A. Borodin, S. A. Cook et N. Pippenger : Parallel computations for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58:113–136, 1983. Cité à la page 94

[12] H. Buhrman, L. Fortnow et T. Thierauf : Nonrelativizing separations. *In IEEE Conference on Computational Complexity*, p. 8–12, 1998. Cité à la page 70, 108

[13] P. Bürgisser : *Completeness and Reduction in Algebraic Complexity Theory*. Num. 7 de Algorithms and Computation in Mathematics. Springer, 2000. Cité à la page 19, 20, 35, 45, 64, 66, 67

[14] P. Bürgisser : The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004. Cité à la page 109

[15] P. Bürgisser : On defining integers in the counting hierarchy and proving lower bounds in algebraic complexity. *In 24th Symposium on Theoretical Aspects of Computer Science*, vol. 4393 de *Lecture Notes in Computer Science*, p. 133–144. Springer-Verlag, 2007. Version longue : Electronic Colloquium on Computational Complexity, Report No. 113 (2006). Cité à la page 29, 31, 32, 41, 43, 44, 45, 46, 47, 64

[16] P. Bürgisser, M. Clausen et M. A. Shokrollahi : *Algebraic Complexity Theory*, vol. 315 de *Grundlehren der mathematischen Wissenschaften*. Springer, 1997. Cité à la page 2, 79, 109

[17] J. F. Canny : Generalized characteristic polynomials. *In Proc. ISSAC'88*, p. 293–299, 1988. Cité à la page 62, 63

[18] O. Chapuis et P. Koiran : Saturation and stability in the theory of computation over the reals. *Annals of Pure and Applied Logic*, 99:1–49, 1999. Cité à la page 24

[19] P. Charbit, E. Jeandel, P. Koiran, S. Perifel et S. Thomassé : Finding a vector orthogonal to roughly half a collection of vectors. *Journal of complexity*, 2006. À paraître. Cité à la page 91, 92

[20] P. Clote et E. Kranakis : *Boolean Functions and Computation Models*. Texts in Theoretical Computer Science (an EATCS Series). Springer, 2002. Cité à la page 98

[21] R. Cole : Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, 1988. Cité à la page 80, 102

[22] Complexity Zoo : `http://qwiki.caltech.edu/wiki/Complexity_Zoo`. Cité à la page 7

[23] F. Cucker et D. Grigoriev : On the power of real Turing machines over binary inputs. *SIAM Journal on Computing*, 26(1):243–254, 1997. Cité à la page 81, 100

[24] W. Eberly : Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989. Cité à la page 97

[25] N. Fichtas, A. Galligo et J. Morgenstern : Precise sequential and parallel complexity bounds for quantifier elimination over algebraically closed fields. *Journal of Pure and Applied Algebra*, 67:1–14, 1990. Cité à la page 56, 57, 79, 80, 81, 100

[26] L. Fortnow et S. Homer : A short history of computational complexity. *Bulletin of the EATCS*, 80:95–133, 2003. Cité à la page 2

[27] H. Fournier et P. Koiran : Are lower bounds easier over the reals? *In Proc. 30th ACM Symposium on Theory of Computing*, p. 507–513, 1998. Cité à la page 39, 50, 79

[28] H. Fournier et P. Koiran : Lower bounds are not easier over the reals : Inside PH. *In Proc. 27th International Colloquium on Automata, Languages and Programming*,

vol. 1853 de *Lecture Notes in Computer Science*, p. 832–843. Springer, 2000. Cité à la page 39, 50, 79

[29] D. Grigoriev : Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5:65–108, 1988. Cité à la page 100

[30] D. Grigoriev : Topological complexity of the range searching. *Journal of Complexity*, 16:50–53, 2000. Cité à la page 6, 91, 92, 99, 101, 108

[31] J. Heintz : Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 24(3):239–277, 1983. Cité à la page 80

[32] J. Heintz et J. Morgenstern : On the intrinsic complexity of elimination theory. *Journal of Complexity*, 9:471–498, 1993. Cité à la page 40

[33] L. A. Hemaspaandra et M. Ogihara : *The Complexity Theory Companion*. EATCS. Springer, 2002. Cité à la page 7

[34] W. Hesse, E. Allender et D. A. M. Barrington : Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002. Cité à la page 94

[35] S. Homer et S. Mocas : Nonuniform lower bounds for exponential time classes. *In Mathematical Foundations of Computer Science, 20th symposium*, vol. 969 de *Lecture Notes in Computer Science*, p. 159–168, 1995. Cité à la page 70

[36] V. Kabanets et R. Impagliazzo : Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Cité à la page 28

[37] E. Kaltofen : Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM*, 35(1):231–264, 1988. Cité à la page 25

[38] R. Kannan : Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982. Cité à la page 70

[39] R. M. Karp et R. J. Lipton : Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982. Cité à la page 10, 15, 40, 72

[40] P. Koiran : Computing over the reals with addition and order. *Theoretical Computer Science*, 133(1):35–48, 1994. Cité à la page 50

[41] P. Koiran : Randomized and deterministic algorithms for the dimension of algebraic varieties. *In Proc. 38th IEEE Symposium on Foundations of Computer Science*, p. 36–45, 1997. Cité à la page 83, 84

[42] P. Koiran : Circuits versus trees in algebraic complexity. *In 17th Symposium on Theoretical Aspects of Computer Science*, p. 32–52, 2000. Cité à la page 79, 83, 108

[43] P. Koiran : Valiant's model and the cost of computing integers. *Computational Complexity*, 13:131–146, 2004. Cité à la page 45, 64

[44] P. Koiran et S. Perifel : Valiant's model : from exponential sums to exponential products. *In Mathematical Foundations of Computer Science, 31st symposium*, vol. 4162 de *Lecture Notes in Computer Science*, p. 596–607. Springer-Verlag, 2006. Cité à la page 41

[45] P. Koiran et S. Perifel : The complexity of two problems on arithmetic circuits. *À paraître dans Theoretical Computer Science*, 2007. Cité à la page 26

[46] P. Koiran et S. Perifel : Interpolation in Valiant's theory. *En preparation*, 2007. Cité à la page 41, 109

[47] P. Koiran et S. Perifel : VPSPACE and a transfer theorem over the complex field. *In Mathematical Foundations of Computer Science, 32nd Symposium*, vol. 4708 de *Lecture Notes in Computer Science*, p. 359–370. Springer-Verlag, 2007. Cité à la page 80

[48] P. Koiran et S. Perifel : VPSPACE and a transfer theorem over the reals. *In 24th Symposium on Theoretical Aspects of Computer Science*, vol. 4393 de *Lecture Notes in Computer Science*, p. 417–428. Springer-Verlag, 2007. Cité à la page 58, 59, 64, 91

[49] A. Kolmogorov : Combinatorial foundations of information theory and the calculus of probabilities. *Russian Mathematical Surveys*, 38(4):29–40, 1983. Cité à la page 70, 73

[50] T. Lee et A. Romashchenko : Resource bounded symmetry of information revisited. *Theoretical Computer Science*, 345(2–3):386–405, 2005. Version conférence dans 29th Symposium on the Mathematical Foundations of Computer Science, 2004. Cité à la page 70, 71, 73

[51] M. Li et P. Vitányi : *An introduction to Kolmogorov complexity and its applications*. Graduate texts in computer science. Springer, seconde édn, 1997. Cité à la page 71, 73

[52] R. J. Lipton : Straight-line complexity and integer factorization. *In Proc. First International Symposium on Algorithmic Number Theory*, vol. 877 de *Lecture Notes in Computer Science*, p. 71–79. Springer, 1994. Cité à la page 40

[53] T. J. Long : Strong nondeterministic polynomial-time reducibilities. *Theoretical Computer Science*, 21:1–25, 1982. Cité à la page 26

[54] L. Longpré et O. Watanabe : On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, 1995. Cité à la page 71, 73

[55] E. Lucas : Théorie des fonctions numériques simplement périodiques. *American Journal of Mathematics pure and applied*, 1:184–240 and 289–321, 1878. Cité à la page 34

[56] F. S. Macaulay : Algebraic theory of modular systems. *Cambridge tracts*, 19, 1916. Cité à la page 62

[57] G. Malod : *Polynômes et coefficients*. Thèse de doctorat, Université Claude Bernard Lyon 1, 2003. Cité à la page 4, 13, 19, 21, 26, 29, 30, 31, 32, 35, 49, 66

[58] G. Malod : The complexity of polynomials and their coefficient functions. *In 22nd IEEE Conference on Computational Complexity*, p. 193–204, 2007. Cité à la page 32

[59] G. Malod et N. Portier : Characterizing Valiant's algebraic complexity classes. *In Mathematical Foundations of Computer Science, 31st symposium*, vol. 4162 de *Lecture Notes in Computer Science*, p. 704–716. Springer-Verlag, 2006. Cité à la page 29, 30

[60] K. Meer : A note on a $\mathsf{P} \neq \mathsf{NP}$ result for a restricted class of real machines. *Journal of Complexity*, 8:451–453, 1992. Cité à la page 39

[61] G. L. Miller, V. Ramachandran et E. Kaltofen : Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM Journal on Computing*, 17(4):687–695, 1988. Cité à la page 21, 68, 69

[62] J. Milnor : On Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964. Cité à la page 100

[63] C. H. Papadimitriou : *Computational Complexity*. Addison-Wesley, 1994. Cité à la page 7, 8, 9, 16, 17, 34, 68

[64] S. Perifel : Symmetry of information and nonuniform lower bounds. *In Computer Science in Russia*, vol. 4649 de *Lecture Notes in Computer Science*, p. 315–327. Springer-Verlag, 2007. Cité à la page 58, 64, 70, 73, 108

[65] B. Poizat : *Les petits cailloux*. Aléas, 1995. Cité à la page 14, 23, 50

[66] B. Poizat : Une expression de taille polynomiale en n de la factorielle d'un nombre de n chiffres. *Soumis*, 2007. Cité à la page 57, 59, 61

[67] J. Renegar : On the computational complexity and geometry of the first-order theory of the reals, part 1. *Journal of Symbolic Computation*, 13:255–299, 1992. Cité à la page 56, 57, 100

[68] U. Schöning : *Complexity and structure*, vol. 211 de *Lecture Notes in Computer Science*. Springer-Verlag, 1985. Cité à la page 70

[69] J. T. Schwartz : Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. Cité à la page 25, 36, 43

[70] M. Shub et S. Smale : On the intractability of Hilbert's Nullstellensatz and an algebraic version of "$\mathsf{P} = \mathsf{NP}$". *Duke Mathematical Journal*, 81(1):47–54, 1996. Cité à la page 40, 108

[71] S. Smale : Mathematical problems for the next century. *Mathematical Intelligencer*, 20:7–15, 1998. Cité à la page 40

[72] L. J. Stockmeyer : The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. Cité à la page 8

[73] V. Strassen : Vermeidung von Divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. Cité à la page 109

[74] V. Strassen : Polynomials with rational coefficients which are hard to compute. *SIAM Journal on Computing*, 3(2):128–149, 1974. Cité à la page 79, 109

[75] S. Toda : On the computational power of PP and $\oplus\mathsf{P}$. *In Proc. 30th IEEE Symposium on the Foundations of Computer Science*, p. 514–519, 1989. Cité à la page 9, 28, 33

[76] S. Toda : Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Information and Systems*, E75-D(1):116–124, 1992. Cité à la page 29

[77] S. Toda et M. Ogiwara : Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21:316–328, 1992. Cité à la page 33

[78] J. Torán : Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):752–773, 1991. Cité à la page 27

[79] L. G. Valiant : Completeness classes in algebra. *In Proc. 11th ACM Symposium on Theory of Computing*, p. 249–261, 1979. Cité à la page 20, 28, 45, 66

[80] L. G. Valiant : The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. Cité à la page 28

[81] L. G. Valiant : The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979. Cité à la page 34

[82] L. G. Valiant, S. Skyum, S. J. Berkowitz et C. Rackoff : Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983. Cité à la page 63, 69

[83] H. Vollmer : *Introduction to Circuit Complexity : A Uniform Approach*. Springer-Verlag New York, Inc., 1999. Cité à la page 12

[84] K. W. Wagner : The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986. Cité à la page 10, 27

[85] A. Zvonkin et L. Levin : The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970. Cité à la page 70, 73

Résumé

Dans cette thèse, on s'intéresse essentiellement à des questions concernant les classes de complexité algébrique de problèmes de décision et de problèmes d'évaluation. Plus précisément, nous étudions d'une part, le modèle de Blum, Shub et Smale (BSS) pour reconnaître des langages sur une structure quelconque grâce à des circuits algébriques, et d'autre part, le modèle de Valiant pour calculer des polynômes grâce à des circuits arithmétiques. Nos résultats montrent qu'afin de séparer des classes de complexité, il est moins difficile de travailler sur les problèmes d'évaluation, c'est-à-dire dans le modèle de Valiant. Cela confirme notre intuition que, le modèle étant plus simple (il n'y a pas de portes de tests), les résultats doivent être plus simples à obtenir. En particulier, nous montrons deux résultats de transfert. Le premier concerne les versions algébriques de P et NP : sur un corps de caractéristique nulle, séparer P et NP dans le modèle BSS grâce à des problèmes NP « sans multiplication » implique de séparer P et NP dans le modèle de Valiant. Le second concerne la question « $\mathsf{P} = \mathsf{PSPACE}\,?$ » : après avoir défini un analogue de PSPACE dans les deux modèles algébriques, nous montrons que séparer P de PSPACE, sur $\mathbb{R}$ ou sur $\mathbb{C}$, est moins difficile dans le modèle de Valiant.

Par ailleurs, et plus brièvement, nous étudions également la manipulation par machines de Turing de polynômes donnés par des circuits arithmétiques. En effet, encoder des polynômes de cette façon peut être bien plus court que la liste de tous les monômes. Mais l'étude de deux exemples (calculer le coefficient d'un monôme et calculer le degré d'un polynôme) montre qu'il semble difficile de manipuler des polynômes donnés de cette façon, même lorsqu'ils sont de degré polynomial et que l'on travaille sur un corps de caractéristique non nulle.