Université de Montréal

Preuves interactives classiques

par
Hugue Blier

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Avril, 2006

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Preuves interactives classiques

présenté par:

Hugue Blier

a été évalué par un jury composé des personnes suivantes:

Geňa Hahn
président-rapporteur

Alain Tapp
directeur de recherche

Pierre McKenzie
membre du jury

Mémoire accepté le 30 août 2006

RÉSUMÉ

Le présent mémoire a pour sujet les preuves interactives dans le modèle classique de machines de Turing. Celles-ci se formalisent par différentes classes de complexité où un prouveur, infiniment puissant, tente de convaincre un vérificateur, borné polynomialement en temps, de la véracité d'un énoncé. Nous supposons dans ce mémoire que le lecteur est familier avec les concepts de langages, de classes de complexité et de machines de Turing. Les différents résultats et notions sont présentés dans un ordre chronologique en autant que celui-ci permette un ordre intelligible.

Ainsi sont d'abord présentés les modèles de preuves interactives avec bits aléatoires privés puis publics, chacun avec ses différentes caractéristiques. Les deux modèles sont ensuite comparés. L'apport de l'auteur à cette section est d'avoir uniformisé les différentes preuves souvent présentées dans des contextes et langages différents. C'est aussi de présenter les résultats dans un ordre démontrant bien la progression du domaine et les questions qui la soutendaient.

Mots clés : preuves interactives, Arthur-Merlin, complexité, espace polynomial, hiérarchie polynomiale.

ABSTRACT

The subject of this Master's thesis is interactive proofs in the classical model of Turing machines. These are formalized by complexity classes in which an infinitely powerful prover tries to convince a verifier polynomially bounded in time that a statement is true. We take for granted that the reader is familiar with the concepts of languages, complexity classes, and Turing machines. Every definitions and theorems are given in chronological order unless doing so hinders comprehension.

First, the models of interactive proofs with private and public bits are presented. The two models are then compared. The author's contribution to this section is the standardized presentation of the different proofs and concepts that appeared in the literature using sometimes very different notation and nomenclature.

Keywords: Interactive proof, Arthur-Merlin, complexity, polynomial space, polynomial hierarchy.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES APPENDICES

LISTE DES SIGLES

$\mathcal{AM}$	Arthur Merlin
$\mathcal{BPP}$	Temps probabiliste polynomial avec erreur bornée
	Bounded-Error Probabilistic Polynomial-Time
$\mathcal{IP}$	Temps polynomial interactif
	Interactive Polynomial-Time
$\mathcal{MA}$	Merlin Arthur
$\mathcal{NP}$	Temps polynomial nondéterministe
	Non-Deterministic Polynomial-Time
$\mathcal{P}$	Temps polynomial
	Polynomial-Time
$\mathcal{PH}$	Hiérarchie polynomiale
	Polynomial-Time Hierarchy
$\mathcal{PP}$	Temps polynomial probabiliste
	Probabilistic Polynomial-Time
$\mathcal{PPSPACE}$	Espace polynomial probabiliste
	Probabilistic Polynomial Space
$\mathcal{PSPACE}$	Espace polynomial
	Polynomial Space

$\mathcal{SAPTIME}$	Temps stochastique alternatif polynomial *Stochastic-Alternating Polynomial-Time*
$\mathcal{ZK}$	Preuve à divulgation nulle *Zero-Knowledge*

NOTATION

$co\text{-}\mathcal{A}$	Complément de la classe de complexité $\mathcal{A}$.
$\overline{\mathcal{B}}$	Complément du langage $\mathcal{B}$.
$\mathcal{A} \subseteq \mathcal{B}$	La classe de complexité $\mathcal{A}$ est incluse dans la classe $\mathcal{B}$.
$\mathcal{A} \leq \mathcal{B}$	Le langage $\mathcal{A}$ se réduit au langage $\mathcal{B}$.
$\mathcal{A} = \mathcal{B}$	$\mathcal{A} \subseteq \mathcal{B}$ et $\mathcal{B} \subseteq \mathcal{A}$ où $\mathcal{A}$ et $\mathcal{B}$ sont des classes de complexité.
$\mathcal{A} \equiv \mathcal{B}$	$\mathcal{A} \leq \mathcal{B}$ et $\mathcal{B} \leq \mathcal{A}$ où $\mathcal{A}$ et $\mathcal{B}$ sont des langages.

À mon père.

REMERCIEMENTS

Bien sûr, écrire un mémoire demande beaucoup d'heures de travail. Si habituellement l'on remercie ceux qui nous ont accompagnés durant ces heures, la rédaction d'un mémoire étant un travail solitaire, je voudrais remercier ceux sans qui le travail eut été plus long et ardu : Anne, Julie, Frédéric, Michel et Simon.

Je me dois d'exprimer ma gratitude à mon directeur, Alain Tapp, qui par sa confiance, son aide, son enseignement et son support m'a permis de vivre dans le monde de l'informatique théorique.

Il faut aussi pouvoir vivre pendant les années que dure une maîtrise, et, pour ce, je tiens à souligner l'importante contribution financière de l'Institut canadien de recherches avancées.

Finalement, je tiens à remercier tout spécialement mes parents qui, sans savoir ce que je faisais, m'ont toujours soutenu et sans qui ce travail n'eut jamais été entrepris.

AVANT-PROPOS

Le présent projet est simplement motivé par la richesse de son sujet. Bien qu'il soit relativement jeune, le domaine des preuves interactives a connu beaucoup d'avancées importantes que j'ai pu découvrir au fil de mes lectures. J'ai pu aussi apprécier la difficulté qu'il y a à aborder ce sujet tant les façons de présenter chacune de ses facettes divergent. Il m'a aussi semblé qu'aujourd'hui les résultats semblent plus difficiles à atteindre, signe de maturité du domaine. Le temps était donc bien choisi pour faire une présentation fluide et consistante, dans un langage uniforme et intelligible, des différents résultats et de leur preuve. Si ce mémoire atteint au moins en partie ce but, je serai satisfait.

CHAPITRE 1

INTRODUCTION

À la base même des mathématiques se trouve l'idée de preuve. C'est-à-dire par une suite d'applications de règles simples de logique appliquées à un ensemble d'axiomes arriver à démontrer une certaine assertion. Une preuve peut être facile à obtenir ou difficile, en supposant qu'elle existe. La complexité, en tant que discipline, aborde ce problème ; en fait, de façon plus générale, la difficulté de résoudre certains problèmes. Mais qu'arrive-t-il lorsqu'une preuve est déjà connue ?

En fait, il suffit d'y croire. Si l'on peut obtenir une preuve courte, il suffit de dire à une personne incrédule - et qui voudrait donc la vérifier - comment cette preuve se fait. Mais si cette preuve est trop longue pour qui voudrait la vérifier, y a-t-il de l'espoir ?

Voilà le problème des preuves interactives. On suppose un prouveur étant infiniment puissant, et un vérificateur ayant, lui, une quantité de ressources limitée (typiquement un temps polynomial en la taille du problème). Est-il possible que le vérificateur soit convaincu de la véracité d'une affirmation même s'il ne peut communiquer que selon sa limitation de ressources avec un prouveur en qui il n'a pas confiance ? Étonnamment, la réponse est oui.

En fait, une preuve interactive n'est jamais une "vraie" preuve. Le vérificateur, à la fin de l'interaction ne peut jamais que dire : "C'est probablement vrai". La raison en est simple et tous les protocoles reposent sensiblement sur la même idée : le vérificateur pose certaines questions au prouveur ; celles-ci doivent avoir la propriété que si l'affirmation est vrai, le prouveur pourra y répondre correctement avec une grande probabilité. Par contre, si elle est fausse, il ne doit pouvoir y répondre de façon satifaisante qu'avec une faible probabilité.

1.1 Structure du mémoire

Au chapitre 2, nous présentons $\mathcal{P}$, $\mathcal{BPP}$ et $\mathcal{NP}$, des classes à la base des preuves interactives. Nous y présentons la définition de $\mathcal{NP}$ vue comme une preuve interactive à un message.

Au chapitre 3, nous présentons quelques caractéristiques importantes de l'interaction et les utilisons pour définir la classe $\mathcal{IP}$. Aussi montrons-nous l'équivalence entre plusieurs définitions de cette classe. Le chapitre suivant se concentre sur le modèle de preuves interactives à bits publics. Nous y présentons aussi un résultat intéressant, propre à cette classe.

Les deux modèles sont montrés équivalents au chapitre 5 et équivalents à la classe $\mathcal{PSPACE}$ au chapitre 6. Finalement, le chapitre 7 introduit deux caractéristiques très instructives de cette dernière.

1.2 Contribution de l'auteur

La contribution de l'auteur est d'abord dans la présentation des différents résultats. Ainsi avons-nous pris soin d'uniformiser la notation et le langage utilisés dans les différents énoncés et preuves.

N'ayant pas ici de contrainte d'espace, une contribution importante dans le rendu des résultats a été de remplir les manques, de compléter les preuves et d'expliciter ces "trivial" si nombreux dans la littérature. Parfois même, la preuve lui est complètement attribuable (ex. théorème 3). Ainsi, le lecteur pourra apprécier que nous n'ayons pas seulement recopié les preuves mais que nous ayons tenté de les rendre plus claires.

1.3 Travaux reliés

La facture de ce mémoire est purement théorique tout comme son sujet. Aussi étudions-nous ici ces classes de complexité pour elles-mêmes. Il est toutefois possible de relier ceci à différents sujets et problèmes.

1. $\mathcal{P}$ est-il égal à $\mathcal{PSPACE}$?

   C'est là une des questions les plus fondamentales de la complexité. Différentes classes, classiques et quantiques, sont prouvées comme étant entre ces deux classes. Il serait donc possible, en les différenciant, de répondre à cette question [Sto87].

   Aussi, $\mathcal{P}$, $\mathcal{NP}$ et $\mathcal{PSPACE}$ peuvent être considérés comme des classes de preuves interactives. Ainsi, il serait peut-être possible de montrer qu'elles sont équivalentes ou non en utilisant le formalisme des preuves interactives.

2. Preuves à divulgation nulle.

   Les preuves interactives sont directements liées à la classe des preuves à divulgation nulle. Les protocoles de cette classe ont des applications très intéressantes, en particulier dans le domaine de la sécurité de la communication et de l'authentification. Le lecteur intéressé aux preuves interactives sous cet angle est invité à lire [Rot01, Gol01]

3. Classer des problèmes

   $\mathcal{IP}$ et $\mathcal{AM}$ offrent une hiérarchie entre $\mathcal{P}$ et $\mathcal{PSPACE}$ qui permet directement et intuitivement de classer certains problèmes. Nous n'avons qu'à penser à l'origine de la classe $\mathcal{AM}$ où Babai [Bab85] tentait de déterminer la difficulté, entre autre, de l'appartenance à un groupe étant donné la liste de ses générateurs.

CHAPITRE 2

NOTIONS PRÉLIMINAIRES

2.1 Introduction

Dans sa vie, un mathématicien, lorsqu'il est devant un énoncé, est amené à deux choses. D'une part, il peut chercher à prouver cet énoncé. Pour beaucoup de théorèmes et de problèmes, il est possible de trouver une preuve, une solution, de façon efficace. Pour faire ceci, aucune interaction n'est nécessaire.

D'autre part, si cette preuve est en sa possession, il peut chercher à la vérifier. Il existe, en effet, des problèmes pour lesquels on ne sait pas trouver de solution de façon efficace ; c'est-à-dire que l'on ne connaît pas d'algorithme permettant de résoudre le problème en temps polynomial. Mais pour certains d'entre eux, on connaît une façon efficace de vérifier une preuve. C'est-à-dire qu'étant donné la preuve d'une certaine assertion, il est possible de vérifier sa validité.

Dans ce chapitre, nous présentons les deux classes de complexité traduisant ces deux contextes ainsi qu'une troisième permettant d'introduire le concept de classe probabiliste.

2.2 $\mathcal{P}$

La première classe à regarder est celle dont les énoncés peuvent être vérifiés en temps polynomial. Formellement, cette classe se nomme $\mathcal{P}$ et se définit ainsi [Sip97] :

Définition 1 (Classe $\mathcal{P}$). *Un langage L est dit dans la classe $\mathcal{P}$ (Polynomial-Time) s'il existe une machine de Turing déterministe et un polynôme $T(n)$ tels que pour tous mots de L de longueur n la machine s'arrête dans l'état acceptant en au plus $T(n)$ étapes.*

Par exemple, il est possible de décider si une formule booléenne est dans $2\mathcal{SAT}$ en temps polynomial et donc $2\mathcal{SAT} \in \mathcal{P}$.

Définition 2 ($2\mathcal{SAT}$). *Ensemble des formules booléennes exprimées en forme normale conjonctive avec 2 littéraux par clause telles que pour chacune d'elle il existe une affectation de variables telle qu'elle s'évalue à vrai.*

Théorème 1. $2\mathcal{SAT} \in \mathcal{P}$

Preuve :

Pour prouver ceci, il suffit de donner un algorithme qui fonctionne en temps polynomial permettant, pour tout ϕ une expression booléenne en forme normale conjonctive avec 2 littéraux par clause, de décider si ϕ est satisfaisable. Cet algorithme provient de [dem05].

Commençons par remarquer que dans une clause $(x \vee y)$, si l'une ou l'autre des variables est fausse, alors l'autre se doit d'être vraie pour que l'expression entière soit satisfaite. Nous disons donc que la valeur de vérité de la première se propage à la seconde. Ce n'est pas le cas si une variable a comme valeur de vérité vrai. L'algorithme suivant fera usage de cette propagation.

Au début du protocole, aucune variable n'est marquée puis, pour chacune des variables non-marquées, nous suivons le protocole suivant :

1. Mettre la variable à vrai.
2. Propager la valeur de la variable puisqu'il peut y avoir négation de cette variable dans l'expression.

3. S'il y a une contradiction, mettre la variable à faux et propager à nouveau la valeur de la variable.
4. S'il y a encore contradiction, rejeter.
5. Marquer les variables concernées par la propagation à leurs valeurs de vérité attribuées.

Tel que dit précédemment, si, dans une clause, l'une des variables est fausse, alors l'autre doit être vraie pour que l'expression soit satisfaite. Aussi est-il suffisant de faire cette propagation car, n'y ayant que deux variables à chaque clause, chaque variable fausse ne laisse qu'un choix de propagation. C'est en particulier la raison pour laquelle il n'est pas nécessaire de reconsidérer des variables déjà traitées, ce que l'on nomme "backtracking".

La propagation de la valeur de vérité d'une variable x_i à l'itération i se fait sur chacune des variables (au plus). Pour chaque variable, il faut parcourir chacune des clauses. Puisque l'on ne fait la propagation qu'une fois par variable à l'intérieur d'une itération, chaque itération se fait en temps

$$\text{Nombre de variables} \times \text{Nombre de clauses}$$

Puisqu'il y a une itération par variable, le protocole se fait en temps polynomial. □

2.3 $\mathcal{NP}$

Il y a aussi des classes de problèmes que l'on croit ne pas pouvoir résoudre en temps polynomial par une machine de Turing déterministe. L'une d'elle se nomme $\mathcal{NP}$ [Sip97].

Définition 3 (Classe $\mathcal{NP}$). *$\mathcal{NP}$ (Non-Deterministic Polynomial-Time) est la classe des langages ayant un vérificateur polynomial. C'est-à-dire que $L \in \mathcal{NP}$ si et seulement s'il existe un polynôme $p(x)$ et une machine de Turing déterministe V (le vérificateur) telle que pour tout mot $m \in L$ il existe un certificat c de taille polynomiale tel que*

$$V \text{ accepte } \langle m, c \rangle \text{ en temps polynomial } p(|m|)$$

Aussi, si $x \notin L$ alors il n'existe pas de tel c.

Nous disons ici "que l'on croit" parce qu'il n'exite pas à ce jour de preuve sur l'équivalence des classes $\mathcal{P}$ et $\mathcal{NP}$. Par contre, $\mathcal{P} \subseteq \mathcal{NP}$ et la conjecture répandue est que cette inclusion soit stricte.

Voilà la première classe de problèmes que nous considérerons comme étant potentiellement plus difficile que la classe des problèmes dont la preuve se trouve en temps polynomial. Le but du présent travail n'est pas de discuter de nondéterminisme et encore moins de tenter de prouver quoi que ce soit sur les rapports entre les classes $\mathcal{P}$ et $\mathcal{NP}$. Cette dernière classe est intéressante en ce qu'elle peut être vue comme la classe de tous les langages ayant une preuve courte, et donc que l'on peut lire en temps polynomial. Voyons un exemple.

Définition 4 ($3\mathcal{SAT}$). *Ensemble des formules booléennes exprimées en forme normale conjonctive avec 3 littéraux par clause telles que pour chacune d'elle il existe une affectation de variables telle qu'elle s'évalue à vrai.*

Théorème 2. $3\mathcal{SAT} \in \mathcal{NP}$

Preuve :

Clairement, l'affectation de chacune des variables est de taille polynomiale en la taille du problème. Elle peut donc constituer le certificat c. Par la définition du langage, $3\mathcal{SAT} \in \mathcal{NP}$ si et seulement s'il existe un tel certificat faisant que l'expression

booléenne s'évalue à vrai. Il est possible à un vérificateur de vérifier chacune des clauses et d'accepter le mot si et seulement si chacune des clauses est vraie, ceci en temps polynomial. □

Ceci a d'intéressant que c peut être considéré comme une preuve courte. Et si l'on veut mettre cette idée sous forme de preuve interactive, il suffit d'imaginer un prouveur donnant c au vérificateur. Il n'est pas possible pour le prouveur de faire accepter au vérificateur un mot qui n'est pas dans le langage puisque si celui-ci n'est pas dans le langage c'est que par définition il n'existe pas de c faisant accepter un vérificateur. Cette idée a pour la première fois été présentée par Cook [Coo71] et, indépendamment, Levin [Lev73].

2.4 $\mathcal{BPP}$

Une autre classe de problèmes potentiellement plus difficiles que ceux dans $\mathcal{P}$ est la classe $\mathcal{BPP}$. Celle-ci repose sur une autre idée : tenter de vérifier soi-même un problème trop difficile, et donc aboutir à une conclusion probabiliste. Nous définirons celle-ci relativement à $\mathcal{NP}$ mais il ne faut pas perdre de vue qu'il n'y pas ici de prouveur.

Définition 5 (Classe $\mathcal{BPP}$). *Classe des langages pour lesquels il existe un vérificateur polynomial tel que :*

1. *si $x \in L$ alors au moins $\frac{2}{3}$ des certificats font accepter le vérificateur.*
2. *si $x \notin L$ alors au plus $\frac{1}{3}$ des certificats font accepter le vérificateur.*

Ainsi, il est possible de choisir des certificats aléatoires et de vérifier chacun d'eux pour obtenir un résultat de plus en plus certain.

CHAPITRE 3

PREUVES INTERACTIVES

3.1 Introduction

Le principe des preuves interactives a pour la première fois été énoncé dans [GMR85]. Goldwasser, Micali et Rackoff ne semblent pas avoir construit la nouvelle classe de complexité qui en découle pour elle-même. Ils cherchaient plutôt à mesurer la quantité d'information contenue dans une preuve et se sont particulièrement intéressés aux preuves n'en contenant aucune. En effet, ils remarquent que dans la preuve d'un théorème il y a beaucoup plus d'information que le seul bit nécessaire à dire : "oui, c'est vrai". Ils donnent pour exemple une preuve de résiduosité quadratique. Ce langage est dans $\mathcal{NP}$ puisqu'il suffit au vérificateur de connaître la factorisation du module. Mais la factorisation d'un nombre est de l'information donnée. Ils donnent ainsi un système de preuve interactive ne donnant au vérificateur aucune information et définissent ainsi la classe $\mathcal{ZK}$ (Zero Knowledge).

Bien qu'ils se soient particulièrement intéressés à cette dernière classe, nous nous intéresserons plutôt à leur définition de preuves interactives. Nous verrons aussi d'autres définitions qui puissent sembler tout aussi valables. Nous montrerons qu'elles sont en fait toutes équivalentes.

3.2 Notions préalables

Il est possible de formaliser l'idée d'un langage ayant une preuve courte, donc dans $\mathcal{NP}$, par l'interaction de deux machines de Turing déterministes, l'une dont le temps est borné polynomialement et l'autre infiniment puissante. La première sera nommée V, pour vérificateur, et l'autre P, pour prouveur. Le vérificateur possède un ruban de lecture pour l'entrée, un ruban de travail (lecture et écriture)

et un autre ruban C de lecture seule. Le prouveur possède, lui, un ruban de lecture pour l'entrée, un ruban de travail et un autre ruban C d'écriture seulement. Le ruban C est pour la communication entre les deux. Le vérificateur et le prouveur interagissent sur le même problème si

1. Leur ruban d'entrée est le même.
2. Leur ruban C est le même.

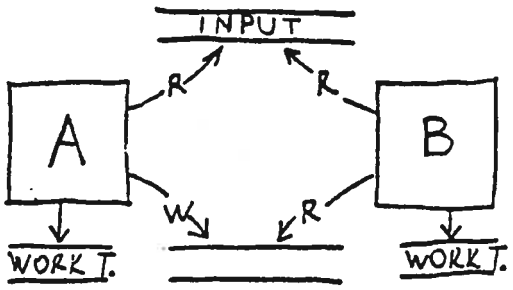

Fig. 1: The *NP* proof-system(*)

FIG. 3.1 – Diagramme original de [GMR85]

Ainsi pour résoudre un problème dans $\mathcal{NP}$, le prouveur lit le problème en entrée, calcule le certificat et l'inscrit sur le ruban de communication. Le vérificateur lit, en plus du problème en entrée, ce certificat et fait les calculs nécessaires en temps polynomial. Si le vérificateur s'arrête en un état acceptant alors il accepte, sinon, il rejette.

La question qui peut venir à l'esprit est : "voilà qui est fait si c est une preuve, mais ne pourrait-il pas plutôt être un indice seulement ?". Étant un indice seulement, il n'est pas conclusif et aussi plusieurs indices peuvent être nécessaires. Goldwasser, Micali et Rackoff présentent les choses avec une merveilleuse image. Ils comparent $\mathcal{NP}$ avec les preuves qui peuvent être écrites dans un livre et, comme professeurs, se disent que lorsqu'ils présentent une preuve dans un cours, ils prennent tout

avantage à l'interaction avec les étudiants : aux points cruciaux ceux-ci peuvent poser des questions et obtenir les réponses.

"Writing down a proof that can be checked by everybody without interaction is a much harder task. In some sense, because one has to answer in advance all possible questions." [GMR85]

On peut donc généraliser la formalisation de $\mathcal{NP}$ pour donner lieu à la classe $\mathcal{IP}$. On définit premièrement le vérificateur comme une machine de Turing polynomiale ayant un ruban de lecture pour l'entrée, un ruban de bits aléatoires en lecture seule, un ruban de travail et deux rubans C_1 et C_2, l'un étant d'écriture et l'autre de lecture. Puis le prouveur comme une machine de Turing ayant un ruban de lecture pour l'entrée, un ruban de bits aléatoires en lecture seule, un ruban de travail et deux rubans C_1 et C_2, l'un étant de lecture et l'autre d'écriture.

Définition 6 (Interaction). *On dit qu'un prouveur et un vérificateur interagissent si*

1. *Leur ruban d'entrée est le même.*
2. *Le ruban d'écriture C_i de l'un est le ruban de lecture de l'autre.*

De façon intuitive, cela définit la communication entre deux machines de Turing à l'aide de deux rubans, l'un servant à envoyer des messages et l'autre à en recevoir. Puisque le vérificateur est borné polynomialement en temps, le nombre de questions-réponses et leur longueur doivent être polynomials. En particulier, et de façon triviale, nous voyons que $\mathcal{NP} \subseteq \mathcal{IP}$.

Une notion seulement manque pour totalement généraliser $\mathcal{NP}$, celle de sécurité. En effet, si l'on se souvient de l'introduction, on se souvient d'un problème de confiance entre le prouveur et le vérificateur. C'est un problème essentiel à la per-

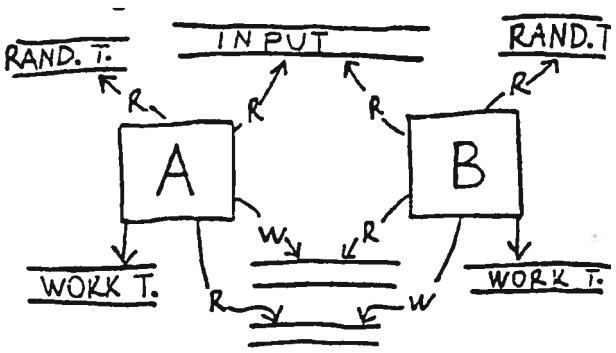

Fig. 2: an interactive pair of Turing machines

FIG. 3.2 – Diagramme original de [GMR85]

tinence de la classe $\mathcal{IP}$; s'il y avait une totale confiance entre les deux, un simple bit de promesse suffirait comme preuve. Ce manque de confiance ne posait pas de problème pour la classe $\mathcal{NP}$ puisque, par définition, il suffisait qu'il existe un certificat faisant accepter le vérificateur pour que le mot soit dans le langage. Mais pour la classe $\mathcal{IP}$, la question se pose : existe-t-il une interaction faisant accepter le vérificateur alors que le mot n'est pas dans le langage ? Aussi doit-on s'assurer de ne pas refuser de mots qui soient effectivement dans le langage. Ceci se formalise par deux définitions :

Définition 7 (Complétude). *On dit d'un certain algorithme effectué par le vérificateur qu'il a une complétude de $c(|x|)$ pour un certain langage L si $\forall x \in L$ $\exists P$ tel que de l'exécution du protocole résulte l'acceptation du mot x avec probabilité plus grande ou égale à $c(|x|)$.*

Définition 8 (Résilience). *On dit d'un certain algorithme effectué par le vérificateur qu'il a une résilience de $r(|x|)$ pour un certain langage L si $\forall x \notin L$ et $\forall B$ où B est une machine interactive quelconque faisant office de prouveur, de l'exécution du protocole résulte l'acceptation du mot x avec probabilité plus petite ou égale à $r(|x|)$.*

3.3 Définition

Il est donc maintenant possible de définir avec ces deux notions la classe des preuves interactives.

Définition 9 (Classe $\mathcal{IP}$). *Un langage L est dans la classe $\mathcal{IP}$ (Interactive Polynomial-Time) s'il existe un vérificateur borné en temps polynomial tel que*

1. *La complétude est plus grande ou égale à $\frac{2}{3}$*
2. *La résilience est plus petite que $\frac{1}{3}$*

Notons que "Polynomial-Time" réfère à la limitation de ressources du vérificateur. Pour le reste du présent travail, la borne sera toujours celle-ci, nous nous abstiendrons donc de la mentionner explicitement. Il peut tout de même être intéressant d'étudier d'autres bornes pour le vérificateur [FL93].

Comme le lecteur s'en doute, il est possible de subdiviser la classe $\mathcal{IP}$ en sous-classes selon le nombre de messages.

Définition 10. *Un langage L est dans la classe $\mathcal{IP}[k]$ s'il est dans la classe $\mathcal{IP}$ et s'il existe un algorithme interactif tel que pour tout mot du langage il y a au plus k messages échangés entre le prouveur et le vérificateur.*

Immédiatement, nous pouvons dire qu'à première vue, $\mathcal{NP} \neq \mathcal{IP}[1]$ puisque dans le premier cas, la complétude doit être 1 alors que dans le second, il suffit qu'elle soit de $\frac{2}{3}$; un raisonnement semblable s'applique pour la résilience. Il est par contre important de souligner que ceci ne constitue en rien une preuve que ces deux classes ne soient égales.

La question de savoir si cette classe peut être ramenée à un nombre constant de messages peu importe la taille du problème est un problème ouvert dont nous traiterons au chapitre 7. Aussi considérons-nous de façon générale que le nombre de messages est polynomialement proportionnel à la taille du problème. L'exemple suivant en est toutefois un qui peut être résolu en un nombre constant de messages avec une complétude de 1. Mais définissons d'abord ce qu'est un isomorphisme.

Définition 11. *On dit que deux graphes sont isomorphes s'il existe une bijection f entre les sommets du graphe 1 et ceux du graphes 2 telle que la relation d'adjacence entre les sommets est respectée. C'est-à-dire que pour toute paire de noeuds du graphe 1 x, y*

$$(x, y) \text{ sont adjacents } \Leftrightarrow (f(x), f(y)) \text{ le sont aussi.}$$

Définition 12 (Non-isomorphisme de graphe). $\overline{\mathcal{GI}}$ *est le langage des paires de graphes non isomorphes. C'est à dire que* $\langle G_0, G_1 \rangle \in \overline{\mathcal{GI}} \Leftrightarrow G_0 \ncong G_1$

Théorème 3. $\overline{\mathcal{GI}} \subseteq \mathcal{IP}$

Ce protocole est souvent attribué à [GMW91] bien que nous ayons trouvé une parution plus ancienne dans [GS86]. Ce problème n'a pas de solution connue dans $\mathcal{P}$, ni même dans $\mathcal{NP}$. Pour montrer que ce langage, $\overline{\mathcal{GI}}$, est dans $\mathcal{IP}$, il suffit de donner un protocole avec une complétude et une résilience satisfaisantes.

Entrée : Un encodage de G_0 et G_1.

V : Il lit un bit sur le ruban aléatoire $b = 0, 1$ et effectue une permutation aléatoire, en lisant d'autres bits aléatoires, du graphe G_b. Il transmet ce graphe à P.

P : Envoie à V $r = 0$ si $G_b \equiv G_0$ et $r = 1$ si $G_b \equiv G_1$

V : Il refait le protocole une seconde fois ; si $r = b$ dans les deux répétitions, il accepte, sinon il rejette.

Preuve : Si $\langle G_0, G_1 \rangle \in \overline{\mathcal{GI}}$ alors il existe un prouveur tel que pour toute permutation aléatoire d'un graphe G_b celui-ci pourra avec probabilité 1 répondre r tel que $r = b$: le graphe donné au prouveur étant isomorphe à un seul des deux graphes originaux, le prouveur pourra toujours répondre correctement. Donc $c(|x|) = 1$.
Si $\langle G_0, G_1 \rangle \notin \overline{\mathcal{GI}}$ alors le prouveur ne pourra deviner qu'avec probabilité $\frac{1}{2}$ si $b = 0$ ou si $b = 1$ puisque G_b est isomorphe aux deux : la permutation de l'un ou l'autre des graphes originaux a la même distribution, ne donnant ainsi aucune information au prouveur sur b. Donc $r(|x|) = \frac{1}{2}$. En répétant la vérification deux fois alors $r(|x|) = \frac{1}{4}$ et donc $\overline{\mathcal{GI}} \subseteq \mathcal{IP}$. □

3.4 Définitions

Cet exemple amène une autre question : la définition de $\mathcal{IP}$ n'est-elle pas arbitraire ? Ne faire le protocole précédent qu'une fois aurait donné $c(|x|) = 1$ et $r(|x|) = \frac{1}{2}$. Pourquoi cela ne suffit-il pas ? D'un autre côté, si l'on répétait le protocole un plus grand nombre de fois, on voit qu'exponentiellement rapidement, on pourrait atteindre une résilience arbitrairement petite ; alors pourquoi demander moins ? Le théorème suivant répond à ces questions en montrant que toutes les conditions intuitives de $\mathcal{IP}$ sont équivalentes.

Théorème 4 (Définitions équivalentes de $\mathcal{IP}$). *Les trois définitions suivantes de $\mathcal{IP}$ sont équivalentes [Gol01] :*

1. *Un langage L est dans la classe $\mathcal{IP}$ s'il existe un vérificateur interactif tel que $c(|x|) \geq \frac{2}{3}$ et $r(|x|) \leq \frac{1}{3}$.*
2. *Un langage L a une preuve interactive forte si pour tout polynôme p, il existe un vérificateur interactif tel que $|c(x)| \geq 1 - \frac{1}{2^{p(|x|)}}$ et $r(|x|) \leq \frac{1}{2^{p(|x|)}}$.*
3. *Un langage L a une preuve interactive faible s'il existe un vérificateur interactif et un polynôme p tel que*

$$c(|x|) - r(|x|) \geq \frac{1}{p(x)}$$

De plus, la complétude et la résilience doivent être calculables en temps polynomial.

Preuve :

Tout au long de cette preuve, et par la suite, nous utiliserons les bornes de Chernoff que nous regroupons sous le théorème suivant [Can] :

Théorème 5. *Soit X le nombre d'essais concluants avec, pour chacun des essais, une variable de Poisson indépendante d'espérance μ. Alors,*

1. *pour $\delta \in (0, 1]$:*
$$Pr[X < (1-\delta)\mu] < \left(\frac{e^{\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}$$
2. *pour $\delta \in (0, 1]$:*
$$Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$
3. *pour $\delta \in (0, 1]$:*
$$Pr[X < (1-\delta)\mu] < e^{\frac{-\mu\delta^2}{2}}$$
4. *pour $\delta < 2e - 1$:*
$$Pr[X > (1+\delta)\mu] < e^{\frac{-\mu\delta^2}{4}}$$

Nous montrerons que $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$.

$1 \Rightarrow 2$: Il s'agit ici de répéter le test un nombre suffisamment grand de fois, de façon indépendante, et de conclure par la réponse majoritaire. Il suffira ici de montrer que le nombre de répétitions est polynomial puisque la longueur de chacune l'est aussi.

Complétude : Nous utiliserons donc le théorème 5. Le test échoue s'il y a une minorité de réponses positives. Soit X le nombre de réponses négatives, nous obtenons, en utilisant la borne de Chernoff, que dans le cas qui nous intéresse, $\mu = \frac{2n}{3}$ et $\delta = \frac{1}{4}$.

$$\begin{aligned} Pr[X < \frac{n}{2}] &< \left(\frac{e^{\frac{-1}{4}}}{(\frac{3}{4})^{(\frac{3}{4})}} \right)^{\frac{2n}{3}} \\ &< .98^n \\ &= 2^{-p(n)} \end{aligned}$$

Ainsi la complétude est supérieure à $1 - 2^{-p(n)}$ où n est le nombre de répétitions.

Résilience : De la même façon, nous utiliserons le théorème 5. Le test échoue s'il y a une majorité de réponses positives. Soit X le nombre de réponses positives, nous obtenons :

$$\begin{aligned} Pr[X > \frac{n}{2}] &< \left(\frac{e^{\frac{1}{2}}}{(\frac{3}{2})^{(\frac{3}{2})}} \right)^{\frac{n}{3}} \\ &< .90^n \\ &= 2^{-p(n)} \end{aligned}$$

Ainsi la résilience est inférieure à $2^{-p(n)}$ où n est le nombre de répétitions.

$2 \Rightarrow 3$:

$$\begin{aligned} c(|x|) - r(|x|) &\geq 1 - 2^{-p(x)} - 2^{-p(x)} \\ &\geq 1 - \frac{1}{2^{p'(x)}} \end{aligned}$$

Puisque par hypothèse ceci est vrai pour tout polynôme, il existe un certain polynôme pour construire la borne inférieure. De plus, les bornes de la résilience et de la complétude sont calculables en temps polynomial puisque toutes deux sont des fonctions exponentielles.

$3 \Rightarrow 1$: Notons m la moyenne entre $c(|x|)$ et $r(|x|)$. Notre protocole pour répondre aux exigences de la définition 1 sera de répéter n fois le protocole respectant la définition 3. La conclusion sera d'accepter le mot si et seulement si le nombre de rejet est plus petit que $m \times n$. Nous montrerons que la complétude et la résilience de la définition 1 sont atteignables et ce dans un nombre polynomial de répétitions.

Soit X le nombre de répétions où l'on a accepté.

Pour la complétude, nous avons

$$Pr[X < m \times n] \leq Pr\left[X < n\left(c(|x|) - \frac{1}{2p(x)}\right)\right]$$

puisque $m \leq (c(|x|) - \frac{1}{2p(x)})$.

En utilisant le théorème 5 avec $\mu = nc(|x|)$ et $\delta = \frac{1}{2c(|x|)p(x)}$, nous pouvons établir n de la façon suivante :

$$\begin{aligned} e^{\frac{-\mu\delta^2}{2}} &= e^{-nc(|x|)\frac{1}{8c(|x|)^2p(x)^2}} = \frac{1}{3} \\ &\Rightarrow \quad n = -8ln\left(\frac{1}{3}\right)c(|x|)p(|x|)^2 \end{aligned}$$

Et donc nous voyons que la complétude est atteignable en un nombre polynomial de répétitions.

De manière similaire, nous avons pour la résilience que

$$Pr[X > m \times n] \leq Pr\left[X > n\left(r(|x|) - \frac{1}{2p(x)}\right)\right]$$

puisque $m \geq (r(|x|) + \frac{1}{2p})$.

Encore une fois, avec le théorème 5 où, dans le cas qui nous intéresse $\mu = nr(|x|)$ et $\delta = \frac{1}{2r(|x|)p(x)}$, nous pouvons établir n de la façon suivante :

$$\begin{aligned} e^{\frac{-\mu\delta^2}{4}} &= e^{-nr(|x|)\frac{1}{16r(|x|)^2p(x)^2}} = \frac{1}{3} \\ &\Rightarrow n = -16ln\left(\frac{1}{3}\right) r(|x|)p(|x|)^2 \end{aligned}$$

Et donc nous voyons que la résilience est atteignable en un nombre polynomial de répétitions. De plus, puisque $r(|x|)$ et $c(|x|)$ sont calculables en temps polynomial, il est possible d'établir quel doit être ce n en prenant le maximum des deux n précédemment calculés. □

Dans le même ordre d'idée, on peut noter [GMS87], qui montre que pour tout langage dans $\mathcal{IP}$ il existe une preuve interactive avec une complétude égale à 1.

CHAPITRE 4

ARTHUR ET MERLIN

4.1 Introduction

De l'exemple de $\overline{\mathcal{GI}}$ du chapitre précédent, il semble que pouvoir garder ses bits aléatoires secrets soit essentiel pour le vérificateur ; celui-ci pose une colle au prouveur et il est évident que si le prouveur connaissait l'indice du graphe qui lui est soumis, il réussirait à tout coup, même si les graphes étaient isomorphes.

Pourtant, nous présenterons ici un modèle de preuves interactives où le vérificateur n'a aucun secret pour le prouveur. Cette classe peut être définie semblablement à la classe $\mathcal{IP}$. Nous pourrons définir à l'intérieur de celle-ci une hiérarchie de classes mais démontrerons que cette hiérarchie "s'écroule".

4.2 Définition

Parallèlement à l'élaboration de $\mathcal{IP}$, Babai a développé un autre type de preuve interactive afin de résoudre un autre problème : prouver l'ordre d'un groupe dont on connait une liste de générateurs et l'appartenance d'un élément à un groupe. Il tente en fait de résoudre ces problèmes dans une classe qui soit à peine plus grande que $\mathcal{NP}$. Dans son article [Bab85], Babai prévoit lui-même l'importance de ce type de preuve : "The tools we introduce seem interesting in their own right". Nous la regarderons, en effet, ici pour elle-même.

L'idée de ce protocole est d'avoir un prouveur, nommé Merlin, qui soit infiniment puissant et un vérificateur, nommé Arthur, qui soit borné polynomialement en temps. Encore une fois, le but de Merlin est de convaincre Arthur que $x \in L$ même si ce dernier ne lui fait pas confiance. La différence entre $\mathcal{IP}$ et cette classe est dans

la restriction qu'a Arthur sur ce qu'il peut envoyer comme message à Merlin. En fait il ne peut envoyer que des bits aléatoires non biaisés.

Définition 13 (Arthur vs. Merlin). *Un langage L est dit dans $\mathcal{AM}[poly]$ s'il existe une machine A bornée polynomialement en temps tel qu'en échangeant un nombre polynomial de messages où le premier provient d'Arthur :*
1 - La complétude est plus grande ou égale à $\frac{2}{3}$
2 - La résilience est plus petite que $\frac{1}{3}$
L'interaction d'Arthur à Merlin est limitée à des bits aléatoires $b = \{(0, \frac{1}{2}), (1, \frac{1}{2})\}$.

Si le nombre de messages est impair, alors Arthur utilise un algorithme probabiliste pour rendre sa décision étant donné la trace jusqu'à maintenant obtenue ; sinon, il utilise un algorithme déterministe.

La classe $\mathcal{MA}[poly]$ peut aussi être définie de façon similaire où le premier message provient de Merlin. Une notation équivalente à la spécification du k pour un nombre constant est d'écrire en alternance les lettres $\mathcal{A}$ et $\mathcal{M}$ k fois. Par exemple, $\mathcal{MA}[4] = \mathcal{MAMA}$. Bien sûr, lorsque la suite se termine par un $\mathcal{A}$, ceci ne signifie pas qu'Arthur envoie un message qui reste sans réponse, mais qu'Arthur utilise un algorithme probabiliste après avoir obtenu le dernier message de Merlin alors qu'il agit de façon déterministe autrement. On voit que $\mathcal{AM}[1] = \mathcal{A} = \mathcal{BPP}$ et que $\mathcal{MA}[1] = \mathcal{M} = \mathcal{NP}$. Il y a aussi certaines inclusions triviales telles que $\mathcal{M} \subseteq \mathcal{MA} \subseteq \mathcal{MAM} \ldots$ On peut aussi voir clairement que, vue la restriction sur les messages d'Arthur, $\mathcal{AM}[poly] \subseteq \mathcal{IP}$. En fait ceci semble si restrictif qu'est née la conjecture que l'inclusion soit stricte.

4.3 Écroulement de la hiérarchie

Cette idée de rendre les bits aléatoires publiques a pour la première fois été énoncée dans [Pap83, Pap85], mais nous y reviendrons. La seule différence par rapport à $\mathcal{IP}$, la restriction d'Arthur sur ses messages, fait en sorte qu'il a été possible à Babai de montrer des propriétés intéressantes de sa hiérarchie. La plus intéressante, et surprenante, étant la suivante :

Théorème 6 (Écroulement de la hiérarchie). *Pour toute constante $k \geq 2$,*

$$\mathcal{AM}[2] = \mathcal{AM}[k]$$

Preuve : L'idée de la preuve est de prendre une conversation Arthur-Merlin de k étapes et d'interchanger l'ordre afin qu'Arthur dise tout ce qu'il a à dire au départ puis que Merlin réponde lui aussi d'un coup. Par exemple une conversation $A_1 - M_1 - A_2$ deviendrait $(A_1, A_2) - M_1$. Intuitivement, on voit que ceci confère un avantage à Merlin du fait qu'il ait connaissance de toute la teneur du message d'Arthur avant même de se commettre à un seul mot d'information lui-même. Pour compenser ceci, plusieurs discussions se feront en parallèle et l'on forcera Merlin à une unique première réponse pour toutes les conversations. La preuve sera présentée de façon similaire à celle de l'article [Bab85] et commence donc par 4 lemmes.

Définissons d'abord formellement le protocole entre Arthur et Merlin de la façon suivante : Soit D_i l'ensemble des messages possibles pour l'un des deux participants au message i, et f une fonction ayant pour domaine le produit cartésien des ensembles D et pour image, $[0, 1]$ telle qu'elle représente la probabilité qu'a Merlin de convaincre Arthur. Il est à noter que si la conversation se termine par un message de Merlin (i.e. $\ldots A_i M_j$) alors la fonction n'a comme image que $\{0,1\}$. Par contre, si la conversation se termine par des calculs d'Arthur (i.e. $\ldots M_i A_j$) alors il est possible pour Arthur de prendre une décision probabiliste et alors la fonc-

tion f a une image continue. Afin d'exprimer une situation de manière succincte, nous adopterons la notation suivante : nous noterons par exemple la probabilité que Merlin a de convaincre Arthur sur une entrée donnée dans un protocole $\mathcal{MA}[4]$ par $Mx_1Ax_2Mx_3Ax_4f(x_1,x_2,x_3,x_4)$ où Mx_1 signifie que Merlin choisit la première variable, notée x_1, Ax_2 signifie qu'Arthur choisit la deuxième variable, notée x_2 et ainsi de suite.

Lemme 1 (Lemme de permutation). *Switching Lemma 3.2 [Bab85]. Soit X et Y deux ensembles non vides et $g(x,y)$ une fonction non négative. Alors, pour $x \in X$ et $y \in Y$,*

$$AyMx\ g(x,y) \leq |X|MxAy\ g(x,y)$$

Preuve : Définissons d'abord la fonction $x(y)$ qui est le choix maximisant les chances de succès de Merlin en réponse au message d'Arthur dans le protocole $\mathcal{AM}[2]$. Définissons aussi une partition de l'ensemble Y, noté $Y(x)$ ainsi : $Y(x) = \{y \in Y | x(y) = x\}$ c'est-à-dire les questions d'Arthur pour lesquelles le choix optimal de Merlin est x. Finalement notons $\psi = MxAyg(x,y)$.

Nous avons donc que $\forall x$

$$\begin{aligned}
\sum_{y \in Y(x)} g(x(y),y) &= \sum_{y \in Y(x)} g(x,y) \\
&\leq \sum_{y \in Y} g(x,y) && \text{Puisque l'image de } g \text{ est positive.} \\
&= |Y|Ayg(x,y) \\
&\leq |Y|\psi && \text{Puisqu'alors } x \text{ est le choix optimal de Merlin.}
\end{aligned}$$

En utilisant cette inégalité, on arrive à la conclusion.

$$
\begin{aligned}
AyMxg(x,y) &= Ayg(x(y),y) \\
&= \frac{\sum_{y\in Y} g(x(y),y)}{|Y|} && \text{Puisqu'Arthur fait un choix uniforme sur l'ensemble } Y\text{, ceci correspond à la moyenne.} \\
&= \sum_{x\in X} \frac{\sum_{y\in Y(x)} g(x(y),y)}{|Y|} && \text{Même somme en utilisant cette fois la partition.} \\
&\leq \sum_{x\in X} \psi && \text{En utilisant l'inégalité précédente.} \\
&= |X|\psi
\end{aligned}
$$

□

Bien sûr, de simplement changer l'ordre donne un avantage certain à Merlin. Pour compenser cet avantage, plusieurs messages seront envoyés en parallèle par Arthur, et Merlin devra répondre à tous par le même message. Plus formellement, nous définissons une fonction $F(\vec{y},x)$ où $x \in X$ et $\vec{y} = [y_1, y_2, \ldots, y_m]$ avec $y_i \in Y$ et m un entier positif, comme la probabilité que Merlin convainque Arthur pour la majorité des conversations.

La première chose est de s'assurer qu'il soit possible de modifier le protocole original afin de diminuer la probabilité d'erreur. Il est en effet nécessaire à la suite de la preuve de pouvoir se prévaloir d'un protocole original avec une erreur arbitrairement petite.

Lemme 2 (Amplification). *Soit $e < \frac{1}{3}$ la probabilité d'erreur de la fonction $f(x,y)$ et E la probabilité d'erreur de la fonction $f^n(\vec{y},x)$ alors $E < \left(\frac{2\sqrt{2}}{3}\right)^n$ où $f^n(\vec{y},x)$ est la probabilité de faire erreur sur la majorité des n conversations en parallèles.*

Preuve : Nous utiliserons la borne de Chernoff (théorème 5, p.16). Dans le cas qui nous intéresse, $\mu = \frac{n}{3}$ et $\delta = \frac{1}{2}$. Donc,

$$\begin{aligned} Pr\left[X > \frac{1}{2}\right] &< \left(\frac{\sqrt{e}}{\left(\frac{3}{2}\right)^{\frac{3}{2}}}\right)^{\frac{n}{3}} \\ &= \left(\frac{2\sqrt{2}\sqrt{e}}{3\sqrt{3}}\right)^{\frac{n}{3}} \\ &\leq \left(\frac{2\sqrt{2}}{3}\right)^{n} \end{aligned}$$

□

Puis finalement, nous avons besoin de deux lemmes pour borner les chances de succès de Merlin dans le cas particulier où Merlin doit répondre à tous les messages d'Arthur de la même façon.

Lemme 3 (Borne supérieure). *Upper Bond Lemma [Bab85]. Soit* $\psi = MxAyf(x, y)$ *avec* $x \in X$ *et* $y \in Y$, *la probabilité qu'a Merlin de convaincre Arthur dans une conversation* $\mathcal{MA}[2]$, *alors :*

$$A\vec{y}MxF(\vec{y}, x) \leq 2^m |X| \psi^{\frac{m}{2}}$$

Preuve : Soit S un sous-ensemble de taille $\lceil \frac{m}{2} \rceil$. Il est aisé de voir que $F(\vec{y}, x)$ est borné supérieurement par la somme sur tous les S de convaincre Arthur sur toutes les conversations $f(x, y_i)$. Dans le premier cas, bien qu'il ne parle en premier, Merlin ne peut faire mieux qu'une réponse optimale pour la moitié des y du vecteur $\vec{y}$ moyen alors que dans le second cas, il peut fournir une réponse optimale pour chacun des sous-ensembles. Donc,

$$F(\vec{y},x) \leq \sum_{S} \prod_{i \in S} f(x,y_i)$$

Ainsi, pour tout $x \in X$, nous avons que :

$$\begin{aligned} A_{\vec{y}} F(\vec{y},x) &\leq \frac{\sum_{\vec{y} \in Y^m} \sum_S \prod_{i \in S} f(x,y_i)}{|Y|^m} \\ &= \sum_S \frac{\sum_{\vec{y} \in Y^m} \prod_{i \in S} f(x,y_i)}{|Y|^m} \end{aligned}$$

Présentons maintenant le terme $\sum_{\vec{y} \in Y^m} \prod_{i \in S} f(x,y_i)$ sous forme de tableau. Fixons S. Comme ce sous-ensemble d'indices s'applique sur le vecteur $\vec{y}$, nous pourrons y trouver toutes combinaisons des termes de $Y^{|S|}$. De ligne en ligne, ce sont les nombres dans les positions du complément de S qui changent et donc nous obtenons $Y^{m-|S|}$ lignes.

$$\left\{ \begin{array}{l} \underbrace{\underbrace{f(x,y_1)f(x,y_1)\dots f(x,y_1)}_{|S|} + f(x,y_1)f(x,y_1)\dots f(x,y_2) + \cdots + f(x,y_{|Y|})f(x,y_{|Y|})\dots f(x,y_{|Y|})}_{|Y|^{|S|}} \\ \vdots \\ \underbrace{\underbrace{f(x,y_1)f(x,y_1)\dots f(x,y_1)}_{|S|} + f(x,y_1)f(x,y_1)\dots f(x,y_2) + \cdots + f(x,y_{|Y|})f(x,y_{|Y|})\dots f(x,y_{|Y|})}_{|Y|^{|S|}} \end{array} \right. Y^{m-|S|}$$

Il est possible de faire des mises en évidence et donc chaque ligne pourrait être réécrite comme :

$$\sum_{j_1 \in \{1,2,\dots,|Y|\}} \sum_{j_2 \in \{1,2,\dots,|Y|\}} \cdots \sum_{j_{|I|} \in \{1,2,\dots,|Y|\}} f(x,y_{j_1})f(x,y_{j_2})\dots f(x,y_{j_{|S|}})$$

Et il est alors possible d'en extraire des moyennes (ici notées $\overline{f(x,y_j)}$).

$$\sum_{j_1\in\{1,2,\ldots,|Y|\}}\sum_{j_2\in\{1,2,\ldots,|Y|\}}\cdots\sum_{j_{|I|}\in\{1,2,\ldots,|Y|\}} f(x,y_{j_1})f(x,y_{j_2})\ldots f(x,y_{j_{|I|}})$$

$$= \sum_{j_1\in\{1,2,\ldots,|Y|\}}\sum_{j_2\in\{1,2,\ldots,|Y|\}}\cdots\sum_{j_{|I|}\in\{1,2,\ldots,|Y|\}} f(x,y_{j_1})f(x,y_{j_2})\ldots\frac{(x,y_{j_{|I|}})}{|Y|}\times|Y|$$

$$= \sum_{j_1\in\{1,2,\ldots,|Y|\}}\sum_{j_2\in\{1,2,\ldots,|Y|\}}\cdots\sum_{j_{|I|}\in\{1,2,\ldots,|Y|\}} f(x,y_{j_1})f(x,y_{j_2})\ldots\overline{f(x,y_j)}\times|Y|$$

$$= \ldots$$

$$= \overline{f(x,y_j)}^{|S|}\times|Y|^{|S|}$$

Comme la représentation en tableau comptait $|Y|^{m-|S|}$ lignes on arrive à ce qu'il soit égal à :

$$\overline{f(x,y_j)}^{|S|}\times|Y|^{|S|}\times|Y|^{m-|S|} = \overline{f(x,y_j)}^{|S|}\times|Y|^m$$

et donc que

$$\sum_S \frac{\sum_{\vec{y}\in Y^m}\prod_{i\in S} f(x,y_i)}{|Y|^m} = \sum_S \overline{f(x,y_j)}^{|S|}$$

$$= \sum_S [Ayf(x,y)]^{|S|}$$

$$< 2^m\psi^{\frac{m}{2}}$$

Puisqu'il y a moins de 2^m sous-groupes et que le fait que Merlin fasse un choix optimal ne peut qu'améliorer ses chances

Nous avons donc montré que $A\vec{y}F(\vec{y},x) \leq 2^m\psi^{\frac{m}{2}}$ et puisque cette inégalité est valide pour tout x alors : $MxA\vec{y}F(\vec{y},x) \leq 2^m\psi^{\frac{m}{2}}$ En appliquant finalement le lemme de permutation, nous pouvons conclure. □

Lemme 4 (Borne inférieure). *Lower Bond Lemma 3.4 [Bab85]. Soit* $\psi = MxAyf(x,y)$, *la probabilité qu'a Merlin de convaincre Arthur dans une conversation* $\mathcal{MA}[2]$, *alors :*

$$1 - A\vec{y}MxF(\vec{y},x) \leq 2^m(1-\psi)^{\frac{m}{2}}$$

Preuve : Soit $\Psi = A\vec{y}MxF(\vec{y},x)$ et x_0 le choix optimal de Merlin dans une conversation $\mathcal{MA}[2]$. De façon évidente, Merlin sera plus convaincant s'il adapte sa réponse au vecteur de questions d'Arthur (dans la simulation) plutôt que de répondre x_0 invariablement.

$$\Psi \geq A\vec{y}F(\vec{y},x_0)$$

La probabilité que Merlin ne convainque Arthur sur plus de la moitié des conversations selon cette dernière stratégie est, de façon semblable au lemme de la borne supérieure, inférieure à la somme des probabilités sur tous les sous-ensembles de taille $\lceil \frac{m}{2} \rceil$, qu'il ne puisse convaincre Arthur dans aucune des conversations du sous-ensemble. i.e.

$$1 - \Psi \leq 1 - A\vec{y}F(\vec{y},x_0) \leq \sum_S (1-\psi)^{|S|} \leq 2^m(1-\psi)^{\frac{m}{2}}$$

□

Montrons maintenant comment simuler un protocole $\mathcal{AM}[k]$ par un protocole $\mathcal{AM}[2]$. Soit le protocole original dont le domaine est $D_1 \times D_2 \times, \ldots D_k$ où $k \geq 3$. Il s'agira d'abord de permuter le deuxième et troisième message tel que dans le lemme de permutation afin de construire une simulation dont le domaine sera $(D_1 \times D_3^m) \times D_2 \times D_4^m \times \cdots \times D_k^m$ pour un certain m.

Il faut maintenant montrer que la simulation est adéquate dans le sens où

1. Celui de Merlin et Arthur qui a l'avantage dans le protocole original, le conserve dans le protocole modifié.
2. L'erreur ne croît pas substantiellement.
3. Le domaine du protocole ne croît pas substentiellement.

Théorème 7 (Équivalence d'une simulation pour une seule permutation). *Soit* $m = 3 \times \max(6, \lceil \log |D_2| \rceil)$ *et supposons que l'erreur du protocole original soit strictement plus petite que* $\frac{1}{18}$. *Alors*

1. *celui de Merlin et Arthur qui a l'avantage dans le protocole original, le conserve dans le protocole modifié ;*
2. E, *l'erreur de la simulation, et* e, *l'erreur du protocole original, sont tels que* $E \leq 9e$;
3. $T \in O(t^2)$ *en autant que* $t > 20$ *où* T *est la taille de la simulation, et* t, *la taille du protocole original.*

Preuve : Montrons d'une part 3 : Soit $T = \log(dom(F))$ et $t = \log(dom(f)) = \log(|D_1|) + \log(|D_2|) + \cdots + \log(|D_k|)$. Souvenons-nous que le domaine de la fonction F est plus petit que m fois celui de la fonction f. Or, puisque $m = 3 \times \lceil \log |D_2| \rceil \in O(t)$ pour $t > 20$ il s'ensuit directement que $T \in O(t^2)$.

D'autre part, soit ϵ la probabilité d'erreur du protocole. En visualisant les interactions entre Arthur et Merlin comme formant un arbre, on voit qu'aucune probabilité n'est changée si l'arbre est tronqué après 3 messages, prenant la somme des probabilités de chacune des branches pour les valeurs de probabilité des feuilles du nouvel arbre ; le domaine du protocole original étant maintenant $D_1 \times D_2 \times D_3$. Une fois le premier message envoyé, x, il reste un seul protocole dont le domaine est $D_2 \times D_3$ et la simulation possible, construite telle que dans les quatre lemmes

précédents, dont le domaine est lui $D_3^m \times D_2$.

Soit $u(x)$ et $U(x)$ l'erreur dans le protocole original et dans la simulation respectivement étant donné x. Alors, pour la résilience,

$$u(x) = MyAxf(y,z) \qquad\qquad U(x) = A\vec{z}MyF(\vec{z},y)$$

et nous pouvons établir la borne suivante :

$$\begin{aligned} U(x) &\leq 2^m |D_2| u(x)^{\frac{m}{2}} && \text{Par le lemme de la borne supérieure} \\ &= 2^{3n} |D_2| u(x)^{\frac{3n}{m}} && m = 3n \\ &\leq 2^{4n} u(x)^{\frac{3n}{2}} && n \geq \log |D_2| \Rightarrow 2^n \geq |D_2| \end{aligned}$$

Et pour la complétude,

$$u(x) = 1 - MyAxf(y,z) \qquad\qquad U(x) = 1 - A\vec{z}MyF(\vec{z},y)$$

et nous pouvons établir la borne suivante :

$$\begin{aligned} U(x) &\leq 2^m u(x)^{\frac{m}{2}} && \text{Par le lemme de la borne inférieure} \\ &= 2^{3n} u(x)^{\frac{3n}{2}} && m = 3n \\ &\leq 2^{4n} u(x)^{\frac{3n}{2}} \end{aligned}$$

Dans les deux cas, on voit que $U(x) \leq u(x)$ si $u(x) \leq \frac{1}{8}$. Si ce n'est pas le cas, on voit que pour un x uniformément distribué :

$$Prob\left[u(x) > \frac{1}{8}\right] < 8E(u(x)) = 8\epsilon$$

et donc la probabilité d'erreur dans la simulation est :

$$E(U(x)) < E(u(x)) + 8\epsilon = 9\epsilon$$

Nous avons donc prouvé la condition 2. Et puisque nous avons supposé que $\epsilon < \frac{1}{18}$, nous avons prouvé la condition 1.

□

Finalement, nous pouvons conclure la preuve du théorème 4 en utilisant d'abord le lemme d'amplification pour que l'erreur du protocole soit suffisamment petite ($\varepsilon \times 9^{\frac{-k}{2}}$ pour un ε désiré). Puis d'utiliser $\frac{k}{2}$ fois le théorème 6. En utilisant le lemme de permutation un nombre constant de fois, nous obtenons, pour le protocole résultant, une taille polynomiale en la taille du protocole original.

Lemme 5. *Pour toute constante $k \geq 3$*

$$\mathcal{MA}[3] = \mathcal{MA}[k]$$

De façon directe, on voit que $\mathcal{MA}[k] \subseteq \mathcal{AM}[k+1] \subseteq \mathcal{AM}[2] \subseteq \mathcal{MA}[3]$ et donc que la hiérarchie s'écroule aussi pour $\mathcal{MA}[k]$.

□

Il est intéressant de voir que cette preuve ne s'applique qu'à $\mathcal{MA}[k]$ pour une constante k et non pas à $\mathcal{MA}[P]$ pour tout polynôme P. La raison en est que la taille du protocole résultant de la construction précédente serait de taille exponentielle en la taille du protocole original.

Cette preuve, en plus de ne considérer qu'un nombre constant de messages, fait grand usage de ce que le message d'Arthur soit totalement public. Il est en effet

beaucoup plus facile d'imaginer Arthur pouvant envoyer tous ces messages en un seul si ceux-ci peuvent être publics que s'il doit préserver un certain secret sur les calculs qu'il effectue d'un message à l'autre. Il serait donc étonnant que $\mathcal{IP}[k]$ s'écroule aussi pour une certaine constante k. Pourtant, ceci a été démontré vrai par Hastad. Malheureusement, il a été impossible de trouver trace de cette preuve.

De toute façon, comme nous le verrons au chapitre suivant, ce résultat n'est plus pertinent. En effet, il a été montré que la classe $\mathcal{IP}$ est équivalente à $\mathcal{AM}$ ce qui implique que $\mathcal{IP}[k]$ s'effondre lui aussi (à $\mathcal{IP}[2]$). Et il semble que ce soit peu avant ce dernier résultat qu'ait été faite la preuve de Hastad, la rendant dès lors désuète.

CHAPITRE 5

ÉQUIVALENCE ENTRE LES DEUX MODÈLES DE PREUVES INTERACTIVES

5.1 Introduction

La simple différence entre la classe $\mathcal{IP}$ et $\mathcal{AM}[poly]$, que les bits soient ou non publics, semble rendre irrémédiablement plus faible cette dernière. L'exemple de $\overline{\mathcal{GI}}$ ne fait que renforcer cette conviction.

Pourtant, $\mathcal{IP} = \mathcal{AM}[poly]$ [GS86] et ce chapitre est entièrement consacré à ce résultat surprenant. Nous verrons aussi un protocole pour $\overline{\mathcal{GI}}$ qui soit dans $\mathcal{AM}$ puisqu'il doit en exister un.

5.2 Notions préalables et lemmes

Définition 14. *Soit $M_{k,b}$ une matrice booléenne, cette matrice définit une fonction linéaire $h : \Delta^k \rightarrow \Delta^b$ où Δ est un alphabet. Nous dirons que cette fonction est une fonction de "hashage" si $b \leq k$.*

Lemme 6. *[GS86]. Soit $b, k > 0$ et $l > max(b, 8)$ et $C \subseteq \Delta^k$. Soit $H = \{h_1, h_2, \ldots, h_l\}$ l fonctions linéaires aléatoirement choisies selon une distribution uniforme et notons $H(C) = \bigcup_i h_i(C)$. Soit $Z = \{z_1, z_2, \ldots, z_{l^2}\}$, l^2 chaînes éléments de Δ^b. Alors :*

1. *Si $b = 2 + \lceil log|C| \rceil$ alors*
   a)$Pr[|H(C)| \geq \frac{|C|}{l}] \geq 1 - 2^{-l}$
   b) $Pr[H(C) \cap Z \neq \emptyset] \geq 1 - 2^{\frac{-l}{8}}$
2. *a) $|H(C)| \leq l|C|$*
   b) Si pour $d > 0$, $|C| \leq \frac{2^b}{d}$ alors $Pr[H(C) \cap Z \neq \emptyset] \leq \frac{l^3}{d}$

Preuve :

1. a) Notons d'abord que :

$$b = 2 + \lceil log|C| \rceil \Rightarrow 2^b = 2^{2+\lceil log|C| \rceil}$$
$$\Rightarrow 2^b \geq 2^2 \times |C| = 4 \times |C|$$

Notons $(h_i(x))_j$ le bit à la position j dans la chaîne $h_i(x)$. Soit $x, y \in \Delta^k$ alors :

$$Pr[(h_i(x))_j = (h_i(y))_j] = \frac{1}{2}$$
$$\Rightarrow Pr[h_i(x) = h_i(y)] = 2^{-b}$$
$$\Rightarrow Pr[\exists y \in C \text{ tel que } x \neq y \text{ et } h_i(x) = h_i(y)] \leq |C| \times 2^{-b} \leq \frac{1}{4}$$

Et puisque les chances sont de $\frac{1}{4}$ pour chaque i.

$$\Rightarrow Pr[\forall i \leq l, \exists y \in C \text{ tel que } x \neq y \text{ et } h_i(x) = h_i(y)] \leq 4^{-l}$$
$$\Rightarrow Pr[\exists x \in C, \forall i \leq l, \exists y \in C \text{ tel que } x \neq y \text{ et } h_i(x) = h_i(y)]$$
$$\leq |C| \times 4^{-l} \leq \frac{2^b}{4} \times 4^{-l} \leq 2^l \times 2^{-2l-2} \leq 2^{-l}$$

Voyons maintenant que $H(C) < \frac{|C|}{l} \Rightarrow \exists x \in C, \forall i \leq l, \exists y \in C$ tel que $x \neq y$ et $h_i(x) = h_i(y)$. Faisons une preuve par contradiction et supposons que $\nexists x \in C, \forall i \leq l, \exists y \in C$ tel que $x \neq y$ et $h_i(x) = h_i(y)$. Ceci correspond au fait que si l'on divise $|C|$ en $|H(C)|$ groupes l fois alors chaque x sera seul au moins une fois. Au plus, pour chaque l, $|H(C) - 1|$ éléments du domaine peuvent être seuls dans leur groupe. Ceci correspond à ce que le nombre d'éléments esseulés soit plus grand que le domaine :

$$l[|H(C)| - 1] \geq |C| \Rightarrow l|H(C)| \geq |C|$$
$$\Rightarrow |H(C)| \geq \frac{|C|}{l}$$

Ceci est une contradiction avec l'hypothèse que $H(C) < \frac{|C|}{l}$. Ainsi, par le résultat précédent, nous avons établi que $Pr[|H(C)| \geq \frac{|C|}{l}] \geq 1 - 2^{-l}$.

b) Puisque $b = 2 + \lceil \log |C| \rceil$ alors

$$b \leq 3 + \log |C| \Rightarrow 2^b \leq 2^{3+\log |C|}$$
$$\Rightarrow \frac{2^b}{8} \leq |C|$$

Il s'ensuit que, en supposant que $|H(C)| \geq \frac{|C|}{l}$:

$$\frac{|H(C)|}{|\Delta^b|} = \frac{|H(C)|}{2^b} \geq \frac{|C|}{l2^b} \geq \frac{2^b}{8 \times l \times 2^b} = \frac{1}{8l}$$

Ainsi $Pr[H(C) \cap Z = \emptyset] \leq (1 - \frac{1}{8l})^{l^2} + 2^{-l} < 2^{\frac{-l}{8}}$ et donc

$$Pr[H(C) \cap Z \neq \emptyset] \geq 1 - 2^{\frac{-l}{8}}$$

2. a) Clairement, pour l fonctions distinctes, il y a au plus $l|C|$ éléments dans l'image et donc $|H(C)| \leq l|C|$.

b) $\frac{|H(C)|}{\Delta^b} = \frac{|H(C)|}{2^b} \leq \frac{l|C|}{2^b} \leq \frac{l|C|}{d|C|} = \frac{l}{d}$. Ainsi la probabilité pour chaque chaîne de Z d'être dans $H(C)$ est $\frac{l}{d}$. Et donc la probabilité qu'une des l^2 chaînes soit dans $H(C)$ est au plus $\frac{l^3}{d}$.

□

Nous aurons aussi besoin d'un autre lemme, un autre lemme d'amplification différent de celui du chapitre précédent.

Lemme 7 (Lemme d'amplification). *Amplification Lemma [GS86]. Soit $p(n)$ un polynôme. Pour tout vérificateur V il existe un vérificateur V' tel qu'ils reconnaissent le même langage où le premier utilise $g(n)$ messages de longueur $m(n)$ utilisant $l(n)$ bits aléatoires et ayant une probabilité d'erreur de $\frac{1}{3}$ alors que le second utilise $g(n)$ messages de longueur $O(m(n)p(n))$ utilisant $O(l(n)p(n))$ bits aléatoires avec une probabilité d'erreur d'au plus $2^{-p(n)}$.*

Preuve : Il suffit de faire $O(p(n))$ fois le protocole en parallèle et de prendre la majorité. □

En particulier, pour la preuve à venir, nous utiliserons $e \leq l(n)^{-12g^2(n)}$ et supposerons que $l(n) > max(g(n), m(n), 80)$. Pour cela, il suffit de faire en parallèle le protocole suffisamment de fois.

5.3 Preuve

Nous montrerons maintenant le théorème suivant :

Théorème 8. *Soit p un polynôme alors $\mathcal{IP}[p] \subseteq \mathcal{AM}[p+2]$.*

Pour compléter la preuve, nous présenterons, $\forall\langle P, V\rangle$ acceptant un certain langage en p messages, un protocole entre Merlin et Arthur acceptant ce même langage en $p+2$ messages, p étant un polynôme. Très grossièrement, l'idée du protocole est de faire générer la conversation qu'un vérificateur et un prouveur pourraient avoir tel que celle-ci réponde à un certain critère difficile à rencontrer si et seulement si en fait la probabilité que le vérificateur accepte ce mot est plus petite que $\frac{1}{3}$.

Dans la description de ce protocole, nous utiliserons la notation suivante : nous utiliserons $r \in \Sigma^l$ pour une chaîne aléatoire et $s_k = x_1y_1\#x_2y_2\#\ldots\#x_ky_k$ pour la

chaîne générée par le protocole entre Arthur et Merlin après k étapes ; x représente le message du vérificateur et y celui du prouveur. Finalement, nous disons que $(V * P)(w, r)$ acceptent via s si sur entrée w et utilisation de la chaîne aléatoire r, les k premiers messages entre V et P sont s et si la continuation du protocole mène finalement à l'acceptation du mot w.

- étape 0 : Arthur ne fait rien.

  Merlin envoie un nombre b_1 à Arthur. Ce nombre est obtenu de la façon suivante : pour chaque $x \in \Sigma^m$ il crée $\alpha_x = \{r | (V * P)(w, r)$ accepte via $x\}$. C'est-à-dire, pour chacun des messages possibles du vérificateur, chercher le nombre de chaînes aléatoires permettant à celui-ci de conclure de façon positive. Puisque le vérificateur utilise l bits aléatoires dans le protocole, il y a 2^l chaînes aléatoires possibles. Ainsi il est possible de regrouper ces α_x dans l classes $\gamma_1, \ldots, \gamma_l$ tel que $\alpha_x \in \gamma_i$ si $2^{i-1} < |\alpha_i| \leq 2^i$. Soit $\bigcup \gamma_i = \bigcup \{\alpha_x | \alpha_x \in \gamma_i\}$ alors on définit γ_{max} comme étant le γ_i pour lequel $|\bigcup \gamma_i|$ est maximal, c'est-à-dire celui contenant le plus de chaînes aléatoires. Finalement, $b_1 = 2 + \lceil \log |\gamma_{max}| \rceil$.

- étape 1 : Arthur, ayant reçu b_1, choisit l fonctions linéaires $H_1 = \{h_1, h_2, \ldots, h_l\}$, $h_i : \Delta^m \to \Delta^{b_1+1}$ et l^2 chaînes aléatoires $Z_1 = \{z_1, z_2, \ldots, z_{l^2}\} \subseteq \Delta^{b_1+1}$ et les envoie à Merlin.

  Merlin cherche un $x \in H^{-1}(Z)$ tel que $\alpha_x \in \gamma_{max}$ (i.e. le nombre de chaînes aléatoires concluantes est maximal) et envoie à Arthur $x_1 y_1$ où $x_1 = x$ et y_1 est la réponse du prouveur ayant reçu x_1 du vérificateur. Si ce x n'existe pas, il envoie plutôt "échec".

  Aussi, Merlin a en sa possession $s_1 = x_1 y_1$. Pour chaque $x \in \Sigma^m$ il crée $\alpha_x = \{r | (V * P)(w, r)$ accepte via $s_1 \# x\}$. Tout comme à l'étape 0, Merlin crée les chasses $\gamma_1, \ldots, \gamma_l$ et envoie à Arthur $b_2 = 2 + \lceil \log |\gamma_{max}| \rceil$.

- étape $2 \leq j \leq g-1$: Arthur ayant reçu jusqu'à maintenant $b_1, \ldots, b_j$ ainsi que s_{j-1}, il vérifie que $x_{j-1} \in H_{j-1}(Z_{j-1})$, sinon il rejette immédiatement. Il choisit l fonctions linéaires $H_j = \{h_1, h_2, \ldots, h_l\}$, $h_i : \Delta^m \to \Delta^{b_j+1}$ et l^2 chaînes aléatoires $Z_j = \{z_1, z_2, \ldots, z_{l^2}\} \subseteq \Delta^{b_j+1}$ et les envoie à Merlin.
  Merlin cherche un $x \in H_j^{-1}(Z_j)$ tel que $\alpha_x \in \gamma_{max}$ (i.e. le nombre de chaînes aléatoires concluantes est maximal) et envoie à Arthur $x_j y_j$ où $x_j = x$ et y_j est la réponse du prouveur ayant reçu x_j du vérificateur. Si ce x n'existe pas, il envoie plutôt "échec".
  Aussi, Merlin a en sa possession s_j. Pour chaque $x \in \Sigma^m$ créer $\alpha_x = \{r | (V * P)(w, r)$ accepte via $s_j \# x\}$. Tout comme à l'étape 1, Merlin crée les classes $\gamma_1, \ldots, \gamma_l$ et envoie à Arthur $b_{j+1} = 2 + \lceil \log |\gamma_{max}| \rceil$.

- étape g : Arthur a reçu jusqu'à maintenant $b_1, \ldots, b_g$ ainsi que s_{g-1}, il vérifie que $x_{g-1} \in H_{g-1}(Z_{g-1})$, sinon il rejette immédiatement. Il choisit l fonctions linéaires $H_g = \{h_1, h_2, \ldots, h_l\}$, $h_i : \Delta^m \to \Delta^{b_g+1}$ et l^2 chaînes aléatoires $Z_g = \{z_1, z_2, \ldots, z_{l^2}\} \subseteq \Delta^{b_g+1}$ et les envoie à Merlin.
  Merlin cherche un $x \in H_g^{-1}(Z_g)$ tel que $\alpha_x \in \gamma_{max}$ (i.e. le nombre de chaînes aléatoires concluantes est maximal) et envoie à Arthur $x_g y_g$ où $x_g = x$ et y_g est la réponse du prouveur ayant reçu x_g du vérificateur. Si ce x n'existe pas, il envoie plutôt "échec".
  Aussi, Merlin a en sa possession s_g. Soit $\alpha_x = \{r | (V * P)(w, r)$ accepte via $s_g\}$, $b_{g+1} = 2 + \lceil \log |\alpha_g| \rceil$ c'est-à-dire 2 plus le logarithme du nombre de chaînes aléatoires permettant au vérificateur de conclure de façon positive. Merlin envoie aussi ce nombre.

- étape $g+1$: Arthur a reçu jusqu'à maintenant $b_1, \ldots, b_{g+1}$ ainsi que s_g, il vérifie que $x_g \in H_g(Z_g)$, sinon il rejette immédiatement. Il choisit l fonctions linéaires $H_{g+1} = \{h_1, h_2, \ldots, h_l\}$, $h_i : \Delta^m \to \Delta^{b_{g+1}+1}$ et l^2 chaînes aléatoires $Z_{g+1} = \{z_1, z_2, \ldots, z_{l^2}\} \subseteq \Delta^{b_{g+1}+1}$ et les envoie à Merlin.
  S'il existe un $r \in \alpha_{x_g} \cap H_{g+1}^{-1}(Z_{g+1})$ alors Merlin l'envoie à Arthur, sinon il envoie "échec". Il est bon de souligner que si $r \in \alpha_{x_g}$ alors le vérificateur, considérant la conversation s_g, aurait accepté w.
  Finalement, Arthur vérifie que $r \in H_{g+1}^{-1}(Z)$. Il vérifie aussi que $\forall i \leq g-1$ le vérificateur, étant donné r, aurait bien produit le message x_{i+1} étant donné s_i. Il vérifie bien sûr que le vérificateur, étant donné r, accepte le mot w étant donné la conversation s_g et que $\sum b_i \geq l - g \log l$. Si c'est le cas, il accepte, sinon, il rejette.

Prouvons maintenant la complétude de ce protocole. Supposons que $w \in L$, et donc que la probabilité que le vérificateur accepte est au moins de $\frac{2}{3}$, montrons que la probabilité qu'Arthur accepte est elle aussi au moins $\frac{2}{3}$. Étant donné un Merlin honnête, le mot est accepté par Arthur si et seulement si Merlin ne fait jamais échouer le protocole et si $\sum b_i \geq l - g \log l$.

Par le lemme précédant (1b) la probabilité que Merlin réponde "échec" à l'un des rounds est au plus $2^{\frac{-l}{8}}$; donc la probabilité que Merlin fasse échouer le protocole est au plus $g2^{\frac{-l}{8}} << \frac{1}{3}$ puisqu'il y a toujours au moins utilisation d'un bit aléatoire dans chacun des messages du vérificateur.

Il reste maintenant à prouver que $\sum b_i \geq l - g \log l$. Ce qui sera fait par les deux lemmes suivants.

Lemme 8. *Claim 1 [GS86]. Soit α_{x_0} l'ensemble de toutes les chaînes aléatoires r, alors $\forall j \leq g$ c'est-à-dire pour chaque b_j relatif au nombre de messages possibles du vérificateur,*

$$|\alpha_{x_j}| \geq \frac{|\alpha_{x_{j-1}}|}{l2^{b_j}}$$

Preuve : On se souvient que

$$\alpha_x = \{r | (V * P)(w, r) \text{ accepte via } s_j \# x\}$$

On peut considérer l'ensemble des α_x à l'étape j comme une partition de l'ensemble des chaînes aléatoires de l'étape précédente dans le sens où :

$$\alpha_{x_{j-1}} = \bigcup_{x_j} \alpha_{x_j}$$

Il suffit de ne pas permettre que le même r soit dans deux α_x distincts en mettant le plus possible de chaînes dans la classe γ_{max}. Aussi avons-nous que le nombre de r dans γ_{max} est au moins le nombre moyen de r dans une des l classes γ_j :

$$\left| \bigcup \gamma_{max} \right| \geq \frac{|\alpha_{x_{j-1}}|}{l}$$

Comme tous les $\alpha_x \in \gamma_{max}$ contiennent au plus le double d'éléments l'un par rapport à l'autre nous avons que, $\forall \alpha_x \in \gamma_{max}$, $|\alpha_x|$ est au moins la moitié de la moyenne de la norme des $\alpha_x \in \gamma_{max}$. Puisque $\alpha_{x_j} \in \gamma_{max}$ alors :

$$|\alpha_{x_j}| \geq \frac{1}{2} \frac{|\bigcup \gamma_{max}|}{|\gamma_{max}|}$$

Nous avons aussi que, puisque $b_j = 2 + \lceil \log |\gamma_{max}| | \rceil$,

$$2^{b_j} = 2^{2 + \lceil \log |\gamma_{max}| | \rceil} \geq 4 \times |\gamma_{max}| \geq 2|\gamma_{max}|$$

Ce qui finalement nous amène à :

$$|\alpha_{x_j}| \geq \frac{|\bigcup \gamma_{max}|}{2|\gamma_{max}|} \geq \frac{|\bigcup \gamma_{max}|}{2^{b_j}} \geq \frac{|\alpha_{x_{j-1}}|}{l 2^{b_j}}$$

□

Lemme 9. *Claim 2 [GS86].* $\sum b_i \geq l - g \log l$

Preuve : En appliquant récursivement le lemme précédent, nous obtenons que

$$|\alpha_{x_g}| \geq \frac{|\alpha_{x_0}|}{l^g \Pi_{j \leq g} 2^{b_j}}$$

où $|\alpha_{x_0}|$ désigne l'ensemble des chaînes aléatoires. Puisque le vérificateur accepte $w \in L$ alors $|\alpha_{x_0}| \geq \frac{2}{3} \times 2^l$. Ainsi nous obtenons que

$$\begin{aligned}
\log |\alpha_{x_g}| &\geq \log \frac{|\alpha_{x_0}|}{l^g \Pi_{j \leq g} 2^{b_j}} \\
&\geq \log \frac{\frac{2}{3} \times 2^l}{l^g \Pi_{j \leq g} 2^{b_j}} \\
&\geq \log \frac{2^{l-1}}{l^g \Pi_{j \leq g} 2^{b_j}} \\
&= l - 1 - g \log l - \sum_{j \leq g} b_j
\end{aligned}$$

Ce qui signifie que $\sum_{j \leq g} b_j \geq -\log |\alpha_{x_g}| - g \log l + (l-1)$.

Par définition, nous avons que $b_{g+1} = 2 + \lceil \log |\alpha_{x_g}| \rceil > 1 + \log |\alpha_{x_g}|$. Ceci nous amène à :

$$\begin{aligned}
\sum_{j \leq g+1} b_j &= \sum_{j \leq g} b_j + b_{g+1} \\
&\geq -\log |\alpha_{x_g}| - g \log l + (l-1) + 1 + \log |\alpha_{x_g}| \\
&= l - g \log l
\end{aligned}$$

□

Montrons maintenant la résilience. C'est-à-dire qu'en supposant que le vérificateur accepte le mot w avec probabilité e, une petite probabilité, alors la probabilité que Arthur accepte ce mot face à n'importe quel Merlin est plus petite ou égale à $\frac{1}{3}$.

Les trois lemmes suivants montreront dans un premier temps qu'à chaque étape, la probabilité que Merlin arrive à convaincre Arthur de l'appartenance de w au langage est réduite considérablement.

Lemme 10. *Soit $P(s_j)$, pour $j > 0$, la probabilité maximale sur tous les prouveurs possibles que le vérificateur accepte w via s_j et y_x la réponse du prouveur qui maximize $P(s_j x y_x)$ alors*

$$P(s_j) = \sum_x P(s_j x y_x)$$

Preuve : Ceci est évident si l'on considère que $P(s_j)$ est égal au nombre de chaînes aléatoires faisant que ce prouveur optimal pourra convaincre le vérificateur par rapport au nombre total de chaînes possibles. Ainsi,

$$\begin{aligned} P(s_j) &= \frac{\text{Le nombre de chaînes aléatoires faisant que (P,V) accepte via } s_j}{\text{Le nombre de chaînes aléatoires total}} \\ &= \sum_x \frac{\text{Le nombre de chaînes aléatoires faisant que (P,V) accepte via } s_j x y_x}{\text{Le nombre de chaînes aléatoires total}} \\ &= \sum_x P(s_j x y_j) \end{aligned}$$

Dans l'argument précédent, il faut considérer que les x forment une partition des chaînes aléatoires. □

Théorème 9. *Pour $0 \leq j < g$, soit $X_c = \{x : P(s_j x y_x) \geq \frac{P(s_j)}{c}\}$ pour une constante $c > 0$ alors*

$$|X_c| \leq c$$

Preuve : Soit P un ensemble de chaînes. D'un côté, divisons-le en c sous-ensembles de taille $\frac{|P|}{c}$. D'un autre côté, divisons-le par un $|X_c|$, le nombre de sous-ensembles de taille plus grande que $\frac{|P|}{c}$. Ainsi, pour tout $|X_c|$,

$$\frac{|P|}{c} \leq \frac{|P|}{|X_c|} \Rightarrow |X_c| \leq c$$

□

Théorème 10. *Soit E_{j+1}, $j < g$, l'événement : $\exists x \in H^{-1}(Z)$ et $y \in \Delta^m$ tel que*

$$P(s_{j+1}) \geq \frac{P(s_j)}{\frac{2^b}{d}}$$

alors

$$Pr(E_i) \leq \frac{l^3}{d}$$

Preuve : Par définition de y_x, $P(s_j x y_x) \geq P(s_{j+1})$. Ainsi si $P(s_{j+1}) \geq \frac{P(s_j)}{\frac{2^b}{d}}$ alors $P(s_j x y_x) \geq \frac{P(s_j)}{\frac{2^b}{d}}$, ce qui est la définition de $x \in X_c$ pour $c = \frac{2^b}{d}$. Il est donc nécessaire à E_{j+1} que $x \in X_c$; ce l'est aussi que $x \in H^{-1}(Z)$.

$$\begin{aligned} Pr(E_{j+1}) &\leq Pr[\exists x \in X_c \cap H^{-1}(Z)] \\ &= Pr[H(X_c) \cap Z \neq \emptyset] \\ &\leq \frac{l^3}{d} \end{aligned}$$

Par le lemme 5, section 2b.

□

Théorème 11. *Claim 5 [GS86]. Soit E_{g+1} l'événement de l'existence d'un r tel que $2^l P(s_g) \leq \frac{2^b}{d}$ et $(V,P)(r)$ accepte via s_g alors*

$$Pr(E_{g+1}) \leq \frac{l^3}{d}$$

Preuve : Soit $R = \{r : (V,P)(r) \text{ accepte via } s_g\}$, alors

$$|R| = 2^l P(s_g) \leq \frac{2^b}{d}$$

Ainsi $E_{g+1} \Rightarrow \exists r \in R \cap H^{-1}(Z)$ et par le même argument que précédemment, nous arrivons à :

$$\begin{aligned} Pr(E_{g+1}) &\leq Pr[\exists r \in R \cap H^{-1}(Z)] \\ &= Pr[H(X_c) \cap Z \neq \emptyset] \\ &\leq \frac{l^3}{d} \qquad \text{Par le lemme 5, section 2b.} \end{aligned}$$

□

Théorème 12. *Si aucun des E_j pour $j \leq g+1$ ne se produit alors Arthur rejette ; la résilience est donc de $\frac{1}{3}$.*

Preuve : Puisque $Pr(E_j) \leq \frac{l^3}{d}$ alors la probabilité qu'au moins l'un d'entre eux se produise est au plus $(g+1)\frac{l^3}{d}$. En posant $d = 3(g+1)l^3$, nous obtenons directement que

$$Pr[\exists i | E_i \text{ se produit}] \leq \frac{1}{3}$$

Montrons maintenant que si aucun de ces événements ne se produit alors Arthur rejettera. D'abord, pour $i \leq g$ nous avons :

$$\frac{P(s_0)}{\prod_{i \leq g}(\frac{2^{b_i}}{d})} \geq P(s_g)$$

.

D'autre part, si $(V,P)(w,r)$ n'acceptait pas, Arthur, n'accepterait pas non plus. Il ne nous reste plus qu'à considérer l'autre cas où $2^l P(s_g) \geq \frac{2^{b_{g+1}}}{d}$.

Nous pouvons donc combiner les deux équations de la façon suivante :

$$\begin{aligned}
\frac{P(s_0)}{P_{s_g}} &\geq \prod_{i \leq g} \left(\frac{2^{b_i}}{d}\right) \\
\Rightarrow 2^l P(s_0) &\geq \prod_{i \leq g+1} \left(\frac{2^{b_i}}{d}\right) \\
\Rightarrow l + \log P(s_0) &\geq \sum b_i - (g+1)\log d \\
&= \sum b_i - (g+1)[\log(3(g+1)) + 3\log l] \\
&\geq \sum b_i - [3g^2 \log l + 3g^2 + 3g] \\
&\geq \sum b_i - 10g^2 \log l
\end{aligned}$$

Et puisque, par hypothèse, le vérificateur accepte avec une probabilité négligeable,

$$P(s_0) = Pr[\text{Vérificateur accepte}] \leq l^{-12g^2}$$

alors :

$$\sum b_i \leq l - 2g^2 \log l < l - g \log l$$

Ainsi, puisque Arthur n'accepte que si

$$\sum b_i \geq l - g \log l,$$

nous pouvons conclure que si aucun E_j ne se produit, Arthur n'acceptera pas.

□

Comme nous l'avons mentionné au chapitre précédent, $\mathcal{AM}[p] \subseteq \mathcal{IP}[p]$. Nous pouvons donc conclure que

Corollaire 1. $\mathcal{IP} = \mathcal{AM}[poly]$

Et du fait que $\mathcal{AM}[k] = \mathcal{AM}[2]$, nous avons que

Corollaire 2. $\mathcal{IP}[k] = \mathcal{IP}[2]$

5.4 Arthur et Merlin pour le nonisomorphisme de graphes

Ce résultat est étonnant ne serait-ce que par l'exemple du protocole pour $\overline{\mathcal{GI}}$ du chapitre 3. Pourtant, le résultat précédent implique qu'il doit y avoir une preuve interactive où la vérificateur rend publics ses bits aléatoires. Nous présentons donc un protocole dans $\mathcal{AM}[2]$ pour ce langage. Celui-ci reprend les idées du protocole de la preuve précédente.

Exemple 1 ($\overline{\mathcal{GI}} \subseteq \mathcal{AM}$). *Soit n le nombre de sommets des graphes G_0 et G_1, définissons*

$$A(G_0, G_1) = \{(G_i, \phi) | i = 0 \text{ ou } i = 1 \text{ et } \phi \in Aut(G_i)\}$$

$$X = A(G_0, G_1) \times A(G_0, G_1) \times A(G_0, G_1) \times A(G_0, G_1) \times A(G_0, G_1) = A(G_0, G_1)^5$$

et h une transformation linéaire aléatoire de l'encodage de X, $\Delta^{p(n)}$, vers $\Delta^{\lceil \log(2^2 \times (n!)^5) \rceil = k}$ alors le protocole est le suivant :

- *Arthur envoie une fonction de hashage linéaire aléatoire à Merlin.*
- *Merlin trouve un $x \in X$ avec une preuve de l'appartenance qu'il envoie à Arthur.*
- *Finalement, Arthur vérifie la preuve d'appartenance ainsi que $h(x) = 0^k$. Si c'est le cas, il accepte.*

Preuve : Cette preuve provient de [KST93] p. 80. Remarquons tout d'abord que

$$A(G_0, G_1) = \{(G_0, \phi) | \phi \in Aut(G_0)\} \cup \{(G_1, \phi) | \phi \in Aut(G_1)\}$$

et donc que :

$$|A(G_0, G_1)| = \begin{cases} n! & G_0 \equiv G_1 \\ 2 \times n! & G_0 \not\equiv G_1 \end{cases}$$

et donc que :

$$|X| = \begin{cases} (n!)^5 & G_0 \equiv G_1 \\ 2^5 \times (n!)^5 & G_0 \not\equiv G_1 \end{cases}$$

Maintenant, soit S la variable aléatoire représentant le nombre d'éléments de X tels que $h(x) = 0^k$. Comme avec probabilité $\frac{1}{2}$ le bit i de $h(x)$ est zéro alors $Prob[h(x) = 0^k] = 2^{-k}$. Donc $E(S) = |X| \times 2^{-k}$.

Soit R_x la probabilité que $h(x) = 0^k$ alors

$$\begin{aligned} Var(S) &= E(S^2) - E(S)^2 \\ &= E((\sum_x R_x)^2) - E(S)^2 \\ &= E(\sum_x \sum_y R_x R_y) - E(S)^2 \\ &= \sum_x \sum_y E(R_x R_y) - E(S)^2 \end{aligned}$$

Le terme de gauche se divise en deux cas. Si $x = y$ alors $E(R_x R_y) = E(R_x) = 2^{-k}$.

Si $x \neq y$ alors la probabilité qu'à la position i $h(x) = 0$ et $h(y) = 0$ est $\frac{1}{4}$ et donc :

$$\begin{aligned}\sum_x \sum_y E(R_x R_y) - E(S)^2 &= \underbrace{|X| \times (|X| - 1) \times 4^{-k}}_{x \neq y} + \underbrace{|X| \times 2^{-k}}_{x=y} - |X|^2 \times 4^{-k} \\ &= |X| \times (|X| - 1) \times 2^{-2k} + |X| \times 2^{-k} - |X|^2 \times 2^{-2k} \\ &= 2^{-k}|X| \times (1 - 2^{-k})\end{aligned}$$

Ainsi nous voyons que

$$\begin{aligned}G_0 \not\equiv G_1 \Rightarrow Prob[\exists x \in X | h(x) = 0^k] \\ &= 1 - Prob[S = 0] \\ &\geq 1 - Prob[|S - E(S)| \geq E(S)] \\ &\geq 1 - \frac{Var(S)}{E(S)^2} && \text{Par l'inégalité de Chebyshev} \\ &= 1 - \frac{2^{-k}|X| \times (1 - 2^{-k})}{(|X| \times 2^{-k})^2} \\ &= 1 - \frac{2^k - 1}{|X|} \\ &\geq 1 - \frac{2^{\log(2^2 \times (n!)^5)}}{2^5 \times (n!)^5} \\ &= 1 - \frac{1}{2^3} \\ &\geq \frac{2}{3}\end{aligned}$$

$$
\begin{aligned}
G_0 \equiv G_1 \Rightarrow Prob[\exists x \in X | h(x) = 0^k] & \\
&= Prob[S \geq 1] \\
&\leq E(S) \\
&= |X| \times 2^{-k} \\
&\leq \frac{(n!)^5}{2^2 \times (n!)^5} \\
&\leq \frac{1}{3}
\end{aligned}
$$

Nous pouvons donc conclure que $\overline{\mathcal{GI}} \in \mathcal{AM}$

□

CHAPITRE 6

CARACTÉRISATION COMPLÈTE DES PREUVES INTERACTIVES

6.1 Introduction

Jusqu'à maintenant, nous avons parlé de la classe $\mathcal{IP}$, et de son équivalent $\mathcal{AM}[poly]$. Mais la question suivante est de savoir où elle se trouve par rapport aux autres classes, on dit alors que l'on veut caractériser cette classe. Et le but ultime est de caractériser complètement une classe, entre autre, de montrer qu'elle est équivalente à une autre classe. Ce classement est souvent artificiel en ce que la classe qui lui est équivalente n'est elle-même placée que de façon indéfinie par rapport à d'autres classes. C'est nécessairement le cas pour toutes classes entre $\mathcal{P}$ et $\mathcal{PSPACE}$ puisque nous ne savons pas si elles sont équivalentes. En particulier, nous aurions pu considérer $\mathcal{AM}$ comme caractérisé à partir du moment où nous avons montré que $\mathcal{IP} = \mathcal{AM}[poly]$, mais nous ne savons toujours rien sur sa position par rapport à d'autres classes de complexité.

Cette caractérisation est par contre beaucoup plus intéressante lorsqu'elle se fait par rapport à des classes connues, étudiées et proches d'une certaine réalité. C'est ce que nous obtenons dans le cas de $\mathcal{IP}$.

6.2 Notions préalables

Nous introduisons brièvement ici une autre classe de complexité qui, *a priori*, n'a rien à voir avec les preuves interactives.

Définition 15 ($\mathcal{PSPACE}$). *Est inclus dans la classe $\mathcal{PSPACE}$ (Polynomial Space) tout langage pouvant être décidé par une machine de Turing déterministe en espace polynomial.*

Cette classe a comme langage complet QBF où le nombre de variables est polynomial par rapport à la taille de l'entrée.

Définition 16 (Expression booléenne quantifiée). *Une expression booléenne quantifiée (Quantified Boolean Fornula, QBF) est une expression booléenne dont toutes les variables sont assujetties à un quantificateur universel ($\forall$) ou un quantificateur existentiel ($\exists$). Elle s'exprime donc sous la forme*

$$\Upsilon_1 x_1 \Upsilon_2 x_2 \ldots \phi(x_1, x_2, \ldots)$$

où ϕ est une expression booléenne et Υ_i est un quantificateur.

Nous profitons de l'occasion pour introduire aussi une infinité de classes de complexité intuitivement reliées à QBF et donc à $\mathcal{PSPACE}$. Ces classes de la hiérarchie polynomiale sont en fait tous les problèmes pouvant se réduire à une QBF avec un nombre fini d'alternances entre les quantificateurs, peu importe la taille de l'entrée. Clairement, $\mathcal{PH} \subseteq \mathcal{PSPACE}$. La définition suivante utilise un oracle. Lorsque nous écrivons "A avec un oracle pour B" (où A et B sont des classes), nous considérons la classe des langages décidables par une machine de Turing utilisant les ressources disponibles dans A et ayant accès à une procédure en temps constant qui décide de l'appartenance à un langage complet pour B.

Définition 17 (Hiérarchie polynomiale). *La hiérarchie polynomiale (Polynomial Hierarchy, $\mathcal{PH}$) est définie [MS72] de la façon suivante : soit : $\Delta_0 = \Sigma_0 = \Pi_0 = \mathcal{P}$ alors pour $i > 0$*

1. $\Delta_i = \mathcal{P}$ *avec un oracle pour* Σ_{i-1}
2. $\Sigma_i = \mathcal{NP}$ *avec un oracle pour* Σ_{i-1}
3. $\Pi_i =$ *co-*$\mathcal{NP}$ *avec un oracle pour* Σ_{i-1}

$$\mathcal{PH} = \cup\Delta_i = \cup\Sigma_i = \cup\Pi_i$$

Lemme 11. *[BC93] Pour $k \geq 0$, $L \in \Sigma_k$ si et seulement si il existe un langage $A \in \mathcal{P}$ et un polynôme p tel que*

$$m \in L \Leftrightarrow \exists x_1 \forall x_2 \exists x_3 \ldots \Upsilon x_k [\langle m, x_1, x_2, x_3, \ldots x_k \rangle \in A]$$

où les quantificateurs doivent alterner et $|x_i| \leq p(|x|)$.

De manière équivalente, $L \in \Pi_k$ si et seulement si il existe un langage $A \in \mathcal{P}$ et un polynôme p tel que

$$m \in L \Leftrightarrow \forall x_1 \exists x_2 \forall x_3 \ldots \Upsilon x_k [\langle m, x_1, x_2, x_3, \ldots x_k \rangle \in A]$$

où les quantificateurs doivent alterneret $|x_i| \leq p(|x|)$.

6.3 $\mathcal{IP} \subseteq \mathcal{PSPACE}$

Dans le but de caractériser la classe $\mathcal{IP}$, nous commencerons par la plus vieille mention que nous ayons pu trouver d'une preuve de ceci. Il est mentionné dans [GS86] que Paul Feldman aurait prouvé qu'un prouveur dans $\mathcal{IP}$ n'a pas besoin de plus de puissance que $\mathcal{PSPACE}$; ce qui implique directement que $\mathcal{IP} \subseteq \mathcal{PSPACE}$ puisque $\mathcal{P}$ (la puissance du vérificateur) $\subseteq \mathcal{PSPACE}$. Malheureusement, ceci réfère à une communication personnelle dont nous n'avons pu trouver trace.

Nous présenterons donc ici une preuve fort simple.

D'abord, notons qu'une preuve interactive peut être représentée sous forme d'arbre qui, sans perte de généralité, représente que le premier message provient du vérificateur (V_0), la réponse du prouveur (P_1), la question suivante du vérificateur (V_2) et ainsi de suite. Notons qu'il y a une distribution uniforme des différentes questions que peut poser le vérificateur alors que le prouveur ne donnera jamais

que la réponse optimale.

Aussi toutes les feuilles sont au même niveau (en ajoutant des communications bidons si nécessaire). Par définition de $\mathcal{IP}$, la longueur de chaque message est bornée par un polynôme $p_1(x)$. Aussi par définition de $\mathcal{IP}$, le nombre de messages doit être polynomial et donc la profondeur de l'arbre est bornée par $p_2(x)$. Chacune des feuilles de l'arbre est la conclusion de la communication représentée par les sommets du chemin à partir de la racine, c'est-à-dire si le vérificateur accepte ou s'il rejette.

Définissons

$$N_i = \begin{cases} 0 & i = p_2(x) \text{ et le protocole rejette} \\ 1 & i = p_2(x) \text{ et le protocole accepte} \\ max_{m_{i+1}}(N_{i+1}) & i < p_2(x) \text{ et impair} \\ moyenne_{m_{i+1}}(N_{i+1}) & i < p_2(x) \text{ et pair} \end{cases}$$

où m_{i+1} est le message vers le niveau m_{i+1}.

Tel que mentionné précédemment, les questions du vérificateur ont une distribution uniforme alors que le prouveur ne répondra jamais que par la réponse optimale. Il est donc clair que N_0 tel que défini ici représente la probabilité qu'a le prouveur d'accepter le mot. Aussi, nous pouvons calculer N_0 récursivement avec une pile de profondeur polynomiale $p_2(x)$ et chaque élément de la pile a aussi une taille polynomiale. Ainsi il est possible de calculer N_0 en espace polynomial. Il suffit par la suite d'accepter le mot si et seulement si $N_0 \geq \frac{2}{3}$ et donc : $\mathcal{IP} \subseteq \mathcal{PSPACE}$.

□

6.4 $\mathcal{IP} = \mathcal{PSPACE}$

On doit la caractérisation complète de $\mathcal{IP}$ à Shamir [Sha92] qui a présenté une preuve constructive sur la façon de résoudre de façon interactive une QBF, démontrant ainsi que $\mathcal{PSPACE} \subseteq \mathcal{IP}$; combiné au résultat précédent, ceci implique que $\mathcal{IP} = \mathcal{PSPACE}$. Nous présenterons ici cette preuve : d'abord quelques propriétés des expressions booléennes quantifiées utiles pour la réalisation du protocole puis le protocole lui-même et finalement la preuve que ce protocole est bien dans $\mathcal{IP}$.

Définition 18 (QBF simple). *Une expression booléenne quantifiée dans laquelle au plus un quantificateur universel sépare chaque variable de son quantificateur est dite une expression booléenne quantifiée simple.*

Théorème 13 (Transformation polynomiale). *Toute expression booléenne quantifiée de taille n peut être transformée en une expression booléenne quantifiée simple équivalente de taille polynomiale $p(n)$.*

Preuve : Pour faire la transformation, il suffit de renommer toutes les variables après chaque quantificateur universel. Par exemple,

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 Q(x_1, x_2, x_3, x_4, x_5, x_6)$$

devient :

$$\exists x_1^0 \forall x_2^0 \exists x_1^1 (x_1^0 = x_1^1) \wedge \exists x_3^0 \forall x_4 \exists x_1^2 \exists x_2^1 \exists x_3^1 (x_1^2 = x_1^1) \wedge (x_2^1 = x_2^0) \wedge (x_3^0 = x_3^1) \\ \wedge \exists x_5 \forall x_6 Q(x_1^2, x_2^1, x_3^1, x_4, x_5, x_6)$$

Cette construction est simple par définition et équivalente à l'expression originale. Pour ce qui est de la taille, notons que chacune des n variables sont renommées au plus n fois (puisqu'il y a au plus n quantificateurs universels) et donc la construction a au plus n^2 termes. □

6.4.1 Protocole

Maintenant, on doit se rappeler que le protocole a pour but de montrer la véracité d'une expression booléenne quantifiée de façon interactive. Donc, soit le problème original, sous forme d'une expression booléenne quantifiée B, nous construisons d'abord son arithmétisation en effectuant les transformations suivantes :

1) Les quantificateurs universels sont remplacés par $\prod_{x_i \in \{0,1\}}$

2) Les quantificateurs existentiels sont remplacés par $\sum_{x_i \in \{0,1\}}$

3) Les $\vee$ sont remplacés par des additions

4) Les $\wedge$ sont remplacés par des multiplications

5) La négation de chaque expression y_i est remplacée par $1 - y_i$

Par exemple :

$$B = \forall x_1 \exists x_2 x_1 \vee \overline{x_2}$$

$$A = \prod_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} x_1 + (1 - x_2)$$

Puis l'interaction commence :

étape 0 : le prouveur évalue A et envoie cette valeur a au vérificateur. Il lui envoie aussi un nombre premier p de taille polynomiale par rapport à l'entrée. Tous les calculs seront $\mod p$.

étape 1 : le prouveur crée $P_1(x_1)$ en évaluant A et en laissant la variable x_1 libre et envoie P_1 au vérificateur. Le vérificateur s'assure que $a = P_1(0) \overset{+}{\times} P_1(1)$ (l'addition si le quantificateur de la variable x_1 est existentiel et la multiplication s'il est universel). Si c'est le cas, le protocole continue et le vérificateur envoie C_1, un nombre aléatoire entre 0 et p.

étape 2 : le prouveur crée $P_2(x_2)$ en évaluant A de sorte que $x_1 = C_1$ et que x_2 soit une variable libre ; il envoie P_2 au vérificateur. Le vérificateur s'assure que $P_2(0) \overset{+}{\times} P_2(1) = P_1(C_1)$. Si c'est le cas, le protocole continue et le vérificateur

envoie C_2 au prouveur.

...

étape i : le prouveur crée $P_i(x_i)$ en évaluant A de sorte que $x_1 = C_1, x_2 = C_2 \ldots x_{i-1} = C_{i-1}$ et que x_i soit une variable libre ; il envoie P_i au vérificateur. Le vérificateur s'assure que $P_i(0) \overset{+}{\times} P_i(1) = P_{i-1}(C_{i-1})$. Si c'est le cas, le protocole continue et le vérificateur envoie C_i au prouveur.

...

étape d : à la dernière étape, le prouveur crée $P_d(x_d)$ en évaluant A de sorte que $x_1 = C_1, x_2 = C_2 \ldots x_{d-1} = C_{d-1}$ et que x_d soit une variable libre ; il envoie P_d au vérificateur. Le vérificateur s'assure que $P_d(0) \overset{+}{\times} P_d(1) = P_{d-1}(C_{d-1})$. Si c'est le cas, le vérificateur génère C_d et vérifie que $P_d(C(d)) = A(C_1, C_2, \ldots, C_d)$. Si oui, le vérificateur accepte le résultat.

Le protocole peut sembler abstrus et contre-intuitif. C'est pourquoi nous le présentons ici de façon abrégé dans un tableau. Il y a deux colones pour le vérificateur, ceci pour mettre en relief qu'il y a dans le même message d'une part la vérification du message du prouveur et d'autre part le début de l'étape suivante.

Vérificateur	Prouveur	Vérificateur
-	a	$a \neq 0$
-	$P_1(x_1)$	$a = P_1(0) \overset{+}{\times} P_1(1)$
C_1	$P_2(x_2)$	$P_2(0) \overset{+}{\times} P_2(1) = P_1(C_1)$
$\vdots$	$\vdots$	$\vdots$
C_{d-1}	$P_d(x_d)$	$P_d(0) \overset{+}{\times} P_d(1) = P_{d-1}(C_{d-1})$
C_d	-	$P_d(C(d)) = A(C_1, C_2, \ldots, C_d)$

TAB. 6.1 – Interaction dans le protocole de Shamir.

6.4.2 Analyse de l'algorithme

Tout d'abord, le protocole se sert de l'arithmétisation d'une expression booléenne. Il faut donc tout d'abord prouver que son évaluation correspond bien à sa valeur de vérité.

Théorème 14 (Valeur de vérité de l'arithmétisation). *Une équation booléenne dans laquelle toutes les variables sont quantifiées est vraie si et seulement si son arithmétisation est différente de zéro.*

Preuve : Nous prouverons les quatres premières étapes d'arithmétisation par induction sur la taille de l'équation booléenne et la cinquième, directement.

Base : soit x une variable, alors $B(x) = x$ est vrai si et seulement si $x \neq 0$. Ce qui est trivial puisque l'on note 0 pour faux et 1 pour vrai.

Induction 1 : Soit $B(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A(x_1, \ldots, x_n) \neq 0$. Montrons que $\forall x_i \; B(x_1, \ldots, x_n)$ vrai $\Leftrightarrow$ $\prod_{x_i} A(x_1, \ldots, x_n) \neq 0$.

$B_{x_1,\ldots,x_n}$ est vrai pour tout $x_i \Leftrightarrow A(x_1, \ldots, x_n) \neq 0 \; \forall x_i$ (par hypothèse d'induction) et donc $\prod_{x_i} A(x_1, \ldots, x_n) \neq 0$.

Induction 2 : Soit $B(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A(x_1, \ldots, x_n) \neq 0$. Montrons que $\exists x_i$ tel que $B(x_1, \ldots, x_n)$ vrai $\Leftrightarrow$ $\sum_{x_i} A(x_1, \ldots, x_n) \neq 0$.

Il existe un x_i tel que $B(x_1, \ldots, x_n)$ soit vrai $\Leftrightarrow A(x_1, \ldots, x_n) \neq 0$ pour ce x_i, par hypothèse d'induction. Comme aucune des valeurs de $A(x_1, \ldots, x_n)$ n'est négative alors $\sum_{x_i} A(x_1, \ldots, x_n) \neq 0$.

Induction 3 : Soit $B_1(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A_1(x_1, \ldots, x_n) \neq 0$ et $B_2(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A_2(x_1, \ldots, x_n) \neq 0$. Montrons que $B_1(x_1, \ldots, x_n) \vee B_2(x_1, \ldots, x_n) \Leftrightarrow$

$A_1(x_1, \ldots, x_n) + A_1(x_1, \ldots, x_n) \neq 0$.

$B_1(x_1, \ldots, x_n) \vee B_2(x_1, \ldots, x_n)$ est vrai $\Leftrightarrow A_1(x_1, \ldots, x_n) \neq 0$ ou $A_1(x_1, \ldots, x_n) \neq 0$, par hypothèse d'induction. Comme il n'y a pas de valeurs négatives de l'arithmétisation alors ceci est vrai si et seulement si $A_1(x_1, \ldots, x_n) + A_2(x_1, \ldots, x_n) \neq 0$

Induction 4 : Soit $B_1(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A_1(x_1, \ldots, x_n) \neq 0$ et $B_2(x_1, \ldots, x_n)$ vrai pour une certaine assignation de variables $\Leftrightarrow A_2(x_1, \ldots, x_n) \neq 0$. Montrons que $B_1(x_1, \ldots, x_n) \wedge B_2(x_1, \ldots, x_n) \Leftrightarrow A_1(x_1, \ldots, x_n) \times A_2(x_1, \ldots, x_n) \neq 0$.

$B_1(x_1, \ldots, x_n) \wedge B_2(x_1, \ldots, x_n)$ est vrai $\Leftrightarrow A_1(x_1, \ldots, x_n) \neq 0 \times A_2(x_1, \ldots, x_n) \neq 0$ par hypothèse d'induction. Et donc si et seulement si $A_1(x_1, \ldots, x_n) \times A_2(x_1, \ldots, x_n) \neq 0$.

Proposition 5 : Pour les négations, il est possible de faire une preuve directe puisqu'il est possible pour toute expression booléenne B, d'exprimer $\overline{B}$ en ne mettant les négations que sur les variables, ceci en suivant les lois de DeMorgan. Ainsi, il suffit de montrer que $\overline{x}$ est vrai pour une certaine affectation $\Leftrightarrow 1 - x \neq 0$.

$\overline{x}$ est vrai $\Leftrightarrow x$ est faux $\Leftrightarrow x = 0 \Leftrightarrow 1 - x = 1$. □

Dans ce protocole tous les calculs sont faits modulo un certain p premier afin de permettre au vérificateur de faire ses calculs. Il faut donc s'assurer qu'il existe toujours un p de taille polynomiale par rapport à B gardant la valeur de vérité.

Théorème 15. *Soit B une expression booléenne de taille n où toutes les variables sont quantifiées. Alors il existe un nombre premier p de taille polynomiale telle que $A \neq 0 \mod p \Leftrightarrow B$ est vrai où A est l'arithmétisation de B.*

Preuve : Montrons d'abord que la valeur de A est bornée par $O(2^{2^n})$. Soit $m(B')$ la valeur maximale que puisse prendre l'expression booléenne sur toutes les valeurs $\{0,1\}$. Alors :

1. $B' = x$ ou $B' = \overline{x} \Rightarrow m(B') = 1$
2. $B' = B'' \vee B''' \Rightarrow m(B') \leq m(B'') + m(B''')$
3. $B' = B'' \wedge B''' \Rightarrow m(B') \leq m(B'') \times m(B''')$
4. $B' = \exists B'' \Rightarrow m(B') \leq 2m(B'')$
5. $B' = \forall B'' \Rightarrow m(B') \leq m(B'')^2$

Il est facile de voir que c'est le cas 5 qui fait croître le plus rapidement la valeur de l'arithmétisation : elle double à chaque quantificateur universel. Ainsi la valeur de l'arithmétisation est $O(2^{2^n})$.

Ceci étant, il nous est possible de faire une preuve par contradiction. Supposons que $A \neq 0$ et $\nexists p | A \neq 0$ pour un p de taille polynomiale. Si A est égal à zéro modulo tous les p de taille polynomiale alors, par le théorème du reste chinois, A est égal à zéro modulo leur produit. Par le théorème des nombres premiers

$$\pi(x) = \frac{x}{log x} = \frac{2^{p(n)}}{p(n)}$$

et donc leur multiplication est $\Omega(2^{2^{p'(n)}})$ pour un polynôme p' quelconque ce qui contredit que la valeur de A soit $O(2^{2^n})$.
De l'autre côté, si B est faux, alors $A = 0$ modulo tout p. □

Il existe aussi une autre preuve équivalente se basant sur la même idée sinon que la croissance de l'espace nécessaire est contrôlée différemment [She92].

Finalement, pour montrer que les messages sont toujours de taille polynomiale, il faut pouvoir borner le degré du polynôme devant être transmis.

Théorème 16. *Si B est simple, alors le degré de la forme fonctionnelle de A, $q(x_i)$ croît au plus linéairement par rapport à la taille de B.*

Preuve : Soit x_i la variable quantifiée la plus à gauche. Alors chaque sommation sur une autre variable ne change le degré de l'expression booléenne et chaque produit double au plus ce degré. L'expression booléenne quantifiée étant simple, cette variable séparée par au plus un produit sur une autre variable et ainsi le degré de $q(x_i)$ ne peut doubler qu'une fois. □

Théorème 17. *La complétude du protocole est 1 et sa résilience est exponentiellement faible. Donc le protocole est dans IP.*

Preuve :

1. Complétude : Si $x \in L$ alors il existe un p de taille polynomiale tel que l'arithmétisation sera différente de zéro modulo p. Ainsi le prouveur pourra toujours suivre le protocole et convaincre le vérificateur.
2. Résilience : Si $x \notin L$ alors a sera toujours égal à 0. Le prouveur doit donc communiquer $a' \neq a$ et donc transmettre un polynôme erroné pour soutenir cette information (i.e. $P_1(0) \overset{+}{\times} P_1(1) \neq a$).

   Suivont maintenant un raisonnement inductif. À la dernière étape, le vérificateur fait sa vérification en calculant directement A, le prouveur doit donner un polynôme qui concorde avec A (i.e. $P_d(x) \equiv A(C_1, C_2, \ldots, C_{d-1}, x)$). S'il ne le fait pas, par le théorème d'interpolation, les deux polynômes ne concorderont qu'en au plus n points pour un polynôme de degré n ; il n'y a donc qu'une probabilité exponentiellement faible pour que le vérificateur fasse sa vérification à un de ces points.

   Si les deux concordent (i.e. $P_d(x) \equiv A(C_1, C_2, \ldots, C_{d-1}, x)$), alors à l'une des étapes (i) le prouveur doit soumettre un polynôme qui concorde avec

A (i.e. $P_i \equiv A(C_1, C_2, \ldots, C_{i-1}, x)$) alors que le polynome P_{i-1} ne concorde pas avec A (i.e. $P_{i-1} \not\equiv A(C_1, C_2, \ldots, C_{i-2}, x)$). Par le théorème d'interpolation, $P_i(0) \overset{+}{\times} P_i(1) = P_{i-1}(C_{i-1})$ pour au plus n C_{i-1} pour un polynôme de degré n ; il n'y a donc qu'une probabilité exponentiellement faible pour que le vérificateur choisisse l'un de ces C_{i-1}.

□

En terminant, il est intéressant de mentionner aussi que d'autres classes de preuves interactives sont complètement caractérisées par des classes courantes. Mentionnons que la classe $\mathcal{IP}$ à 2 prouveurs est égale à la classe des problèmes décidables par une machine de Turing non déterministe en temps exponentiel [FRS94, BFcL90]. Pour d'autres modèles, le lecteur est invité à regarder [For89]. Le lecteur plus avisé sera intéressé à savoir que la preuve présentée ici ne se relativise pas [All90].

CHAPITRE 7

RÉSULTATS CONNEXES

7.1 Introduction

Nous avons montré au chapitre 4 que la hiérarchie $\mathcal{AM}[k]$ s'écroule pour une constante k. Puis la même chose pour $\mathcal{IP}[k]$ au chapitre 5 en montrant que $\mathcal{IP} = \mathcal{AM}$. Finalement, au chapitre précédent, avons-nous montré que $\mathcal{IP} = \mathcal{PSPACE}$ et présenté une certaine forme de hiérarchie incluse dans $\mathcal{PSPACE}$, soit $\mathcal{PH}$.

Si nous croyons que $\mathcal{IP}$ pour un nombre non constant de messages ne s'effondre pas à un nombre constant de messages, qu'en est-il de $\mathcal{PH}$? En fait, pour l'un comme pour l'autre, nous n'avons pas de preuve ; mais la conviction répandue est que l'une comme l'autre ne s'effondre pas. Par contre, nous pouvons présenter un lien entre l'effondrement potentiel de chacune de ces hiérarchies.

L'équivalence entre $\mathcal{IP}$ et $\mathcal{PSPACE}$ montre aussi cette dernière sous un nouveau jour. Nouveau ? Pas tout à fait. En fait, nous présenterons une autre caractérisation de $\mathcal{PSPACE}$ comme un jeu entre deux participants ; une caractérisation qui est venue historiquement avant $\mathcal{IP}$.

7.2 Effondrement de la hiérarchie

Nous venons de voir que $\mathcal{IP}$, à l'instar de la hiérarchie polynomiale, forme, par sa hiérarchie, une mesure de complexité à l'intérieur de la classe $\mathcal{PSPACE}$. Elle est d'autant plus intéressante du fait qu'elle soit équivalente à cette dernière. Une question alors se pose : pourrait-on réduire la complexité de $\mathcal{PSPACE}$ à un nombre fini d'interactions ? Malheureusement, au moment d'écrire ces lignes, la réponse nous est inconnue, tout comme nous ne savons pas si $\mathcal{PH}$ s'écroule de la

même façon. Le seul consensus est qu'il est plus probable pour l'une comme pour l'autre qu'elle ne s'effondre pas, ceci ayant des conséquences surprenantes [K77].

Nous présentons ici un résultat important : si $\mathcal{IP}$ s'effondrait à un nombre constant d'interactions alors $\mathcal{PH}$ s'effondrerait aussi, renforçant ainsi notre conviction que $\mathcal{IP}$ ne peut être réduit à un nombre constant de messages.

Notons d'abord que Lund, Fortnow, Karloff et Nisan [LFKN92] ont prouvé que *co-*$\mathcal{NP}$ est inclus dans $\mathcal{IP}$. Nous ne présenterons pas ce résultat puisque nous avons déjà établi que $\mathcal{IP} = \mathcal{PSPACE}$. Ceci étant acquis, nous montrerons le résultat suivant : Si *co-*$\mathcal{NP}$ est contenu dant $\mathcal{IP}[k]$ alors $\mathcal{PH}$ s'effondre à son deuxième niveau [BHZ87]. Sachant que *co-*$\mathcal{NP}$ est inclus dans $\mathcal{IP}$, il est clair que si $\mathcal{IP}$ s'effondre alors *co-*$\mathcal{NP}$ est inclus dans $\mathcal{IP}[k]$ et donc que $\mathcal{PH}$ s'effondre. Il est intéressant de mettre en relief que le résultat que nous présentons ici a été démontré dans le but de donner un argument en faveur de cette croyance de l'époque qui était que *co-*$\mathcal{NP}$ n'était pas inclus dans $\mathcal{IP}$. C'est en démontrant l'inverse que Lund, Fortnow, Karloff et Nisan ont augmenté la portée de ce théorème. Ici sera utilisé le fait précédemment démontré que $\mathcal{IP}[k] = \mathcal{AM}[2]$ (puisque, du chapitre 5 et 4 respectivement, $\mathcal{IP}[k] \subseteq \mathcal{AM}[k+2] \subseteq \mathcal{AM}[2] \subseteq \mathcal{IP}[k]$) et nous montrerons donc de façon équivalente le théorème suivant :

Théorème 18. *Si co-*$\mathcal{NP} \subseteq \mathcal{AM}[2]$ *alors* $\mathcal{PH}$ *s'effondre au niveau* Π_2^p.

Preuve : Cette preuve se fera en trois étapes, soit les deux lemmes suivants et la conclusion de la preuve.

Lemme 12 (Lemme d'amplification). *Pour tout langage* L *dans* $\mathcal{AM}[2]$ *et polynôme* q, *il existe un langage* M *dans* $\mathcal{NP}$ *et un polynôme* p *tels que :*

1. $\forall x \in L$, *la fraction des chaînes* y *de longueur* $p(|x|)$ *qui satisfont* $xy \in M$ *est au moins* $1 - 2^{-q|x|}$.
2. $\forall x \notin L$, *la fraction des chaînes* y *de longueur* $p(|x|)$ *qui satisfont* $xy \in M$ *sont au plus* $2^{-q|x|}$.

L'intuition est de considérer y comme la question que pose Arthur à Merlin. Ainsi, puisque x est un mot d'un langage dans $\mathcal{AM}[2]$ alors il existe un grand nombre de questions auxquelles Merlin peut répondre correctement. En incluant cette question dans le problème, on crée un mot xy dans un langage $\mathcal{NP}$. Le même raisonnement tient pour $x \notin L$.

Preuve : Puisque $L \subseteq \mathcal{AM}[2]$ alors il existe un langage $M_0 \in \mathcal{NP}$ et un polynôme p_0 tels que

1. $\forall x \in L$, la fraction des chaînes y de longueur $p_0(|x|)$ qui satisfont $xy \in M_0$ est au moins $\frac{2}{3}$.
2. $\forall x \notin L$, la fraction des chaînes y de longueur $p_0(|x|)$ qui satisfont $xy \in M_0$ sont au plus $\frac{1}{3}$.

Maintenant, amplifions la complétude et la résilience : il suffit pour un même x que le vérificateur pose plusieurs questions y au prouveur. Le vérificateur accepte la preuve si le prouveur répond correctement dans la majorité des cas. Plus formellement, soit M un langage pour un x donné tel que $xy_1y_2 \dots y_{34q(|x|)} \in M$ si et seulement si $xy_i \in L$ pour une majorité de i. La longueur de ces mots sera $p(|x|) = 34q(|x|)p_0(|x|)$.

Pour la complétude, posons X le nombre de y_i tels que $xy_i \in M_0$. Alors, par la borne de Chernoff (théorème 5, p.16) où, dans le cas qui nous intéresse, nous avons $\delta = \frac{1}{4}$ et $\mu = \frac{2n}{3}$, nous obtenons :

$$Pr[X < \frac{1}{2}] < e^{\frac{-2n}{3}\frac{1}{16}\frac{1}{2}} < 2^{\frac{-1.44n}{3\times 16}} = 2^{-.03n}$$

Et donc en posant $n = 34q(|x|)$ nous obtenons que la probabilité d'échec est plus petite que $2^{-q(|x|)}$. Nous pouvons donc conclure que :

1. $\forall x \in L$ la fraction des chaînes $y_1y_2 \dots y_{34q|x|}$ de longueur $p(|x|)$ qui satisfont $xy_1y_2 \dots y_{34q(|x|)} \in M$ est au moins $1 - 2^{-q(|x|)}$.

2. $\forall x \notin L$ la fraction des chaînes $y_1 y_2 \dots y_{34q|x|}$ de longueur $p(|x|)$ qui satisfont $xy_1 y_2 \dots y_{34q(|x|)} \in M$ est au plus $2^{-q(|x|)}$.

Il reste à montrer que $M \subseteq \mathcal{NP}$. Ceci est trivial du fait que M_0 soit dans $\mathcal{NP}$: puisqu'il est possible de fournir un certificat pour chacun des xy_i si et seulement si $xy_i \in M_0$, alors il est possible de fournir une majorité de certificats pour chacun des xy_i si et seulement si $xy_i \in M_0$ pour une majorité de i et donc si et seulement si $xy_1 y_2 \dots y_{34q(|x|)} \in M$. □

Lemme 13. *Si co-$\mathcal{NP}$ est inclus dans $\mathcal{AM}[2]$ alors co-$\mathcal{AM}$ est inclus dans $\mathcal{AM}[2]$.*

Preuve : Soit $L \subseteq$ *co-*$\mathcal{AM}$ alors *co-*$L \subseteq \mathcal{AM}[2]$. Par le lemme précédent il existe un langage *co-*M tel que *co-*$M \subseteq \mathcal{NP}$ et un polynôme p tels que

1. $\forall x \in$ *co-*L, la fraction des chaînes y de longueur $p(|x|)$ qui satisfont $xy \in$ *co-*M, est au moins $\frac{9}{10}$.
2. $\forall x \notin$ *co-*L, la fraction des chaînes y de longueur $p(|x|)$ qui satisfont $xy \in$ *co-*M, est au plus $\frac{1}{10}$.

Et en utilisant le complément de ce langage nous obtenons qu'il existe un langage $M \subseteq$ *co-*$\mathcal{NP}$ tel que pour le même p :

1. $\forall x \in L$, la fraction des chaînes y de longueur $p(|x|)$ qui satisfont $xy \in M$ est au moins $\frac{9}{10}$.
2. $\forall x \notin L$, la fraction des chaînes y de longueur $p(|x|)$ qui satisfont $xy \in M$ est au plus $\frac{1}{10}$.

Par hypothèse, nous avons alors que $M \subseteq \mathcal{AM}[2]$. Nous pouvons appliquer le même raisonnement que précédemment et montrer, en appliquant le lemme d'amplification, qu'il existe un langage N tel que $N \subseteq \mathcal{NP}$ et un polynôme q tels que :

1. $\forall xy \in M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N$ est au moins $\frac{9}{10}$.

2. $\forall xy \notin M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N$ est au plus $\frac{1}{10}$.

Il reste finalement à montrer que $L \subseteq \mathcal{AM}[2]$. Pour ce faire nous montrerons que N est tel que

1. $\forall x \in L$, la fraction des chaînes yz de longueur $p(|x|) + q(|x| + p(|x|))$ qui satisfont $xyz \in N$ est au moins $\frac{2}{3}$.
2. $\forall x \notin L$, la fraction des chaînes yz de longueur $p(|x|) + q(|x| + p(|x|))$ qui satisfont $xyz \in N$ est au plus $\frac{1}{3}$.

Ceci se montre directement du fait que pour $x \in L$ la fraction des y de taille $p(|x|)$ tel que $xy \in M$ est $\frac{9}{10}$ et que pour chacun de ces y la fraction des z tel que $xyz \in N$ est aussi de $\frac{9}{10}$. Ainsi la fraction des yz de taille $p(|x|) + q(|x| + p(|x|))$ tel que $xyz \in N$ est $\left(\frac{9}{10}\right)^2 > \frac{2}{3}$. De façon similaire on conclut que si $x \notin L$ la fraction des yz de taille $p(|x|) + q(|x| + p(|x|))$ tel que $xyz \in N$ est $1 - \left(\frac{9}{10}\right)^2 < \frac{1}{3}$.

Ainsi nous venons de démontrer une complétude et une résilience satisfaisantes du langage L et donc $L \in \mathcal{AM}$ □

Nous pouvons donc finir la preuve du théorème principal qui était : si *co*-$\mathcal{NP}$ est contenu dans $\mathcal{AM}[2]$ alors $\mathcal{PH}$ s'effondre à Π_2^p.

Nous montrerons par induction que $\sum_k \subseteq \mathcal{AM}[2]$. La base est triviale puisque :

$$\sum_1 = \mathcal{NP} \subseteq \mathcal{AM}[2]$$

Notre hypothèse d'induction est $\sum_{k-1} \subseteq \mathcal{AM}[2]$. Ainsi, considérons le langage : $L \in \Sigma_k \equiv \{\phi : \underbrace{\exists y \forall a \exists b \ldots}_{k} \phi(y, a, b, \ldots)\}$ ainsi $x \in L \Leftrightarrow \exists y, p$ tel que $xy \in M$ où y est de taille $p(|x|)$ et

$$M \in \Pi_{k-1} \equiv \{\phi : \underbrace{\forall a \exists b \ldots}_{k-1} \phi(a, b, \ldots)\}$$

Nous pouvons appliquer le raisonnement suivant :

$$\begin{aligned} M \in \Pi_{k-1} \equiv \{\phi : \underbrace{\forall a \exists b \ldots}_{k-1} \phi(a, b, \ldots)\} &\Rightarrow \overline{M} \in \Sigma_{k-1} \equiv \{\phi : \underbrace{\exists a \forall b \ldots}_{k-1} \overline{\phi}(a, b, \ldots)\} \\ &\Rightarrow \overline{M} \in \mathcal{AM}[2] && \text{Par induction.} \\ &\Rightarrow M \in \textit{co}\text{-}\mathcal{AM}[2] \\ &\Rightarrow M \in \mathcal{AM}[2] && \text{Lemme précédent.} \end{aligned}$$

Par définition de $\mathcal{AM}$, on peut en conclure qu'il existe un langage N' et un polynôme q tel que

1. $\forall xy \in M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N'$ est au moins $\frac{2}{3}$.
2. $\forall xy \notin M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N'$ est au plus $\frac{1}{3}$.

En appliquant le premier lemme, il est possible de créer un langage N tel que

1. $\forall xy \in M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N$ est au moins $1 - \frac{1}{3} \times 2^{-p(|x|)}$.
2. $\forall xy \notin M$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xyz \in N$ est au plus $\frac{1}{3} \times 2^{-p(|x|)}$.

Nous pouvons finalement définir N_y tel que $xz \in N_y \Leftrightarrow \exists y$ tel que $xyz \in N$. Par la définition de N, nous voyons que

1. $\forall x \in N$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xz \in N_y$ est au moins $1 - \frac{1}{3} \times 2^{-p(|x|)}$.
2. $\forall x \notin N$, la fraction des chaînes z de longueur $q(|xy|)$ qui satisfont $xz \in N_y$ est au plus $\frac{1}{3}$.

Nous obtenons une résilience de $\frac{1}{3}$ puisqu'il n'existe pas de y tel que $xyz \in N$, alors, pour $x \in L$, pour toute chaîne y il y a au plus $\frac{1}{3} \times 2^{-p(|x|)}$ des chaînes z possibles tel que $xz \in N_y$. Puisqu'il y a $2^{p(|x|)}$ chaînes y, ceci fait au plus $2^{p(|x|)} \times \frac{1}{3} \times 2^{-p(|x|)}$ chaînes z possibles.

Puisque $N \in \mathcal{NP}$ alors N_y l'est aussi ce qui implique que $L \in \mathcal{AM}$.

□

7.3 Autre caractérisation de $\mathcal{PSPACE}$

Nous venons donc de voir que $\mathcal{IP}$ peut être vu comme un jeu à 2 joueurs caractérisant $\mathcal{PSPACE}$. Ce n'est en fait pas le premier. On doit celui-ci à Papadimitriou [Pap83,Pap85] sous la dénomination de $\mathcal{PPSPACE}$ (Probabilistic Polynomial Space) puis $\mathcal{SAPTIME}$ (Stochastic-Alternating Polynomial-Time). De façon semblable à $\mathcal{IP}$ il s'agit de définir un jeu où deux participants s'affrontent mais où, contrairement à $\mathcal{IP}$ l'un des joueurs se désintéresse de la partie, c'est-à-dire qu'il ne tente aucun stratagème pour déjouer le prouveur, il ne fait que jouer de façon aléatoire. Pour ce langage, la résilience et la complétude sont de $\frac{1}{2}$, c'est-à-dire

qu'il suffit d'une majorité de positions gagnantes pour gagner. Il s'agissait ici pour Papadimitriou de formuler un problème de décision avec incertitude comme un jeu contre la nature, celle-ci étant désintéressée.

Si l'on veut formuler ce jeu avec nos deux personnages habituels, on doit noter que c'est le vérificateur qui se désintéresse de la partie et que c'est lorsque le prouveur "gagne" qu'en fait le mot est dans le langage, le prouveur ayant réussi à convaincre le vérificateur.

Définition 19 (Classe $\mathcal{PPSPACE}$). *Un langage L est dans la classe $\mathcal{PPSPACE}$ (Probabilistic $\mathcal{PSPACE}$) s'il existe un prouveur et un vérificateur interagissant tel que*

1 - La complétude est plus grande que $\frac{1}{2}$.

2 - La résilience est plus petite ou égale à $\frac{1}{2}$.

où l'interaction du vérificateur vers le prouveur est limitée à des bits aléatoires $b = (0, \frac{1}{2}), (1, \frac{1}{2})$.

Théorème 19. $\mathcal{PPSPACE} = \mathcal{PSPACE}$

Preuve : Montrons d'abord que $\mathcal{PPSPACE} \subseteq \mathcal{PSPACE}$.

Il existe un arbre représentant le jeu de $\mathcal{PPSPACE}$ où les noeuds de profondeur impaire représentent les choix aléatoires (stochastiques) et les noeuds de profondeur paire, les choix du prouveur. Puisque la profondeur d'un tel arbre est polynomiale, il est possible de parcourir en profondeur l'arbre dans un espace polynomial de sorte qu'à tout noeud existentiel on puisse trouver le sous-arbre (le choix à faire pour le prouveur) ayant le rapport $\frac{\text{feuilles acceptantes}}{\text{feuilles rejetantes}}$ le plus élevé et qu'à tous les noeuds stochastiques, on cumule ce rapport de tous les choix possibles. Ainsi, si le rapport à la racine est plus grand que 1, c'est que le mot est dans le langage.

Puis montrons que $\mathcal{PSPACE} \subseteq \mathcal{PPSPACE}$.

Soit l'arbre A représentant le problème de $\mathcal{PSPACE}$ où seuls sont représentés les sous-arbres des choix optimaux (pour les noeux existentiels). Alors ajoutons un noeud stochastique auquel seront reliés A et A' un arbre identique à l'exception des feuilles : toutes y sont rejetantes sauf une. Considérons ce nouvel arbre en considérant chaque noeud universel comme stochastique. Ce nouvel arbre est dans $\mathcal{PPSPACE}$ si et seulement si A n'avait que des feuilles acceptantes si et seulement si A représente le calcul pour un mot dans $\mathcal{PSPACE}$. □

Nous avons présenté une autre caractérisation de $\mathcal{PSPACE}$ comme un jeu entre deux participants. Il est intéressant de noter que d'autres caractérisations de $\mathcal{PSPACE}$ en tant que jeu existent. Celles-ci sont un peu différentes en ce qu'il faut non plus jouer, mais déterminer le gagnant d'un jeu étant donné une position. Nous référons le lecteur intéressé à [Sch78].

CHAPITRE 8

CONCLUSION

Dans ce mémoire, nous avons fait une revue des principales caractéristiques et définitions des preuves interactives dans un ordre logique, proche de l'ordre historique. Tous ces résultats étaient présents dans la littérature ; leur présentation uniformisée et cohérente est le travail de l'auteur.

Tout d'abord, les classes $\mathcal{P}$, $\mathcal{NP}$ et $\mathcal{BPP}$ ont été introduites dans l'optique des preuves interactives. Puis ont été définis des concepts de base tels que l'interaction, la complétude et la résilience qui ont permis de définir formellement la classe $\mathcal{IP}$. Nous avons aussi montré l'équivalence entre des définitions substantiellement différentes de $\mathcal{IP}$.

Au chapitre 4, nous avons introduit de la même façon, la classe $\mathcal{AM}$ et avons présenté une propriété intéressante de cette classe : l'écroulement de sa hiérarchie pour un nombre constant de messages. Et l'équivalence de $\mathcal{IP}$ et $\mathcal{AM}[poly]$ a été montrée.

Ayant montré les deux classes étudiées jusqu'à maintenant équivalentes, nous les avons complètement caractérisées. Nous avons, en effet, présenté une preuve constructive de l'équivalence entre $\mathcal{IP}$ et $\mathcal{PSPACE}$ après avoir défini cette dernière ainsi que QBF, un langage complet pour celle-ci.

Au chapitre 7 nous avons présenté deux caractéristiques de $\mathcal{PSPACE}$ et donc de $\mathcal{IP}$. La première étant que si $\mathcal{IP}$ s'effondre à un nombre constant de messages, alors $\mathcal{PH}$ s'effondre aussi. Ceci renforce la conviction que $\mathcal{IP}$ ne s'effondre pas, sans toutefois être une preuve. Le second résultat est la définition d'une autre classe, $\mathcal{PPSPACE}$, elle aussi équivalente à $\mathcal{PSPACE}$.

BIBLIOGRAPHIE

[All90] Eric Allender. Oracles versus proof techniques that do not relativize. In *Proceedings of the International Symposium on Algorithms*, 1990.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of Symposium on the Theory of Computation*, number 17, 1985.

[BC93] Daniel Pierre Bovet and Pierluigi Crescenzi. *Introduction to the Theory of Complexity.* Prentice Hall, 1993.

[BFcL90] L. Babai, L. Fortnow, and c. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of Foundations of Computer Science*, 1990.

[BHZ87] R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs. *Information Processing Letters*, 25, 1987.

[Can] J. Canny. Chernoff bounds. Lecture 10 of CS174.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of ACM symposium on Theory of computing*, number 3, pages 151–158, 1971.

[dem05] Démonstration 7 - ift2102, A05.

[FL93] L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity. In *Selected papers of the 8th annual symposium on Theoretical aspects of computer science*, 1993.

[For89] L. Fortnow. *Complexity-Theoretic Aspects of Interactive Proof Systems.* PhD thesis, Massachusetts Institute of Technology, 1989.

[FRS94] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 1994.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive protocols. In *Proceedings of Symposium on Theory of Computing*, number 17, pages 291–304, 1985.

[GMS87] O. Goldreich, Y. Mansour, and M. Sipser. Interactive proof systems : Provers that never fail and random selection. In *Symposium on Foundation of Computer Science*, 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity of all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing machinery*, 38(1) :691–729, 1991.

[Gol01] O. Goldreich. *Foundations of Cryptography.* Cambridge, 2001.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of ACM Symposium on Theory of Computing*, number 18, pages 59–68, 1986.

[K77] Stockmeyer K. The polynomial-time hierarchy. *Theoretical Computer Science*, 3, 1977.

[KST93] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem : its structural complexity.* Birkhäuser, 1993.

[Lev73] A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3) :115–116, 1973.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4) :859–868, 1992.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires expenential space. In *IEEE Symposium on Switching and Automata Theory*, 1972.

[Pap83] C. Papadimitriou. Games against nature. In *Proceeding of IEEE Symposium on Foundations of Computer Science*, 1983.

[Pap85] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31, 1985.

[Rot01] J. Rothe. Some facets of complexity theory and cryptography : A five-lectures tutorial. In *Electronic Colloquium on Computational Complexity, Repeort No 96*, 2001.

[Sch78] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16, 1978.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the Association for Computing Machinery*, 39(4) :869–877, October 1992.

[She92] A. Shen. IP = PSPACE : Simplified proof. *Journal of the Association for Computing Machinery*, 39(4), 1992.

[Sip97] M. Sipser. *Introduction to the Theory of Computation.* PWS, 1997.

[Sto87] L. Stockmeyer. Classifying the computational complexity of problems. *the Journal of Symbolic Logic*, 25(1), 1987.

Annexe I

Diagramme d'inclusion

Voici le graphe d'inclusion de quelques classes de complexité afin de mieux situer celles dont nous avons parlé dans ce travail.

Une classe est incluse dans une autre lorsqu'elle est inférieure à une autre à laquelle elle est reliée par un trait. Si certaines classes ne sont pas reliées par un chemin ascendant, c'est que nous ne savons pas leur rapport, et donc leur hauteur relative dans le graphe n'est pas significative. Par exemple, nous savons que $\mathcal{P} \subseteq \mathcal{NP}$ alors que nous ne connaissons pas la relation entre $\mathcal{PH}$ et $\mathcal{PP}$.

Bien sûr, aucune inclusion n'est stricte puisqu'à ce jour, nous n'avons pas de preuve que $\mathcal{P}$ ne soit équivalent à $\mathcal{PSPACE}$.

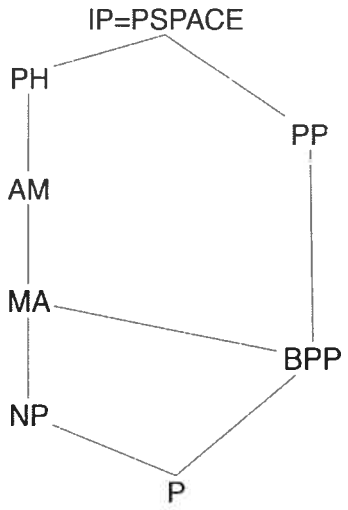

FIG. I.1 – Diagramme d'inclusion