$IP = PSPACE$

Robin Morisset

28 décembre 2009

Table des matières

1 Introduction

Dans ce rapport je vais présenter une classe de complexité, IP.
Cette classe est basée sur la notion de communication entre un prouveur $\mathcal{P}$ et un vérificateur $\mathcal{V}$, et sur la notion de machine de Turing probabiliste. Après une rapide présentation, je montrerais qu'en réalité, ce n'est qu'une autre définition de $PSPACE$ (tout comme on peut re-définir NP comme étant la classe des problème dont une preuve est vérifiable en temps polynômial par une machine de Turing déterministe).

2 Interactive Proofs

2.1 Définition de BPP

Définition 1. *Une machine de Turing probabiliste peut être modélisée comme étant une machine de Turing déterministe normale recevant une entrée supplémentaire : une suite infinie de bits aléatoires.*

Ceci revient à dire qu'elle peut choisir à chaque configuration entre 2 transitions via le tirage d'une pièce à pile ou face.

Définition 2. *On dit qu'un langage $\mathcal{L}$ est dans la classe de complexité BPP si il existe $p < \frac{1}{2}$ et une machine de Turing probabiliste $\mathcal{V}$ avec 2 entrées : un mot x, et le code d'une preuve que $x \in \mathcal{L}$, qui termine toujours et de complexité en temps polynômiale, tels que :*
- *si $x \in \mathcal{L}$ alors $\mathbb{P}[\mathcal{V}(x, \pi) = 0] \leq p$ où π est une preuve valide que $x \in \mathcal{L}$*
- *si $x \notin \mathcal{L}$ alors $\forall \pi, \mathbb{P}[\mathcal{V}(x, \pi) = 1] \leq p$*

2.2 Définition de IP

La classe de complexité IP (dont nous allons montrer qu'elle est égale à $PSPACE$) est une généralisation de BPP.

Définition 3. *Soit $\mathcal{V}$ une machine de Turing probabiliste, terminant toujours et de complexité en temps polynômiale. Soit $\mathcal{P}$ une machine de Turing (sans contrainte sur sa complexité). Supposons que $\mathcal{V}$ peut appeler $\mathcal{P}$ un nombre polynômial de fois pour obtenir des informations. Le résultat d'une exécution de $\mathcal{V}$ sur une entrée x dépend alors de $\mathcal{P}$ et est alors notée $< \mathcal{P}, \mathcal{V} > (x)$.*

Définition 4. *On dit qu'un langage $\mathcal{L}$ est dans la classe de complexité IP si il existe $p < \frac{1}{2}$, une machine de Turing probabiliste $\mathcal{V}$ qui termine toujours et de complexité en temps polynômiale, et une machine de Turing $\mathcal{P}$ tels que :*
- *si $x \in \mathcal{L}$ alors $\mathbb{P}[< \mathcal{P}, \mathcal{V} > (x) = 0] \leq p$*
- *si $x \notin \mathcal{L}$ alors $\forall \mathcal{P}^*, \mathbb{P}[< \mathcal{P}^*, \mathcal{V} > (x) = 1] \leq p$*

Le principal intérêt de cette classe de complexité est qu'en exécutant un grand nombre de fois $< \mathcal{P}, \mathcal{V} > (x)$, on peut prendre p aussi petit qu'on veut.

Théorème 1.

$$IP = PSPACE$$

3 $IP \subseteq PSPACE$

Lemme 1. $IP \subseteq PSPACE$

Cette inclusion est la plus facile des deux à prouver (ou en tout cas celle dont la démonstration est la moins astucieuse).

Démonstration. Soit $\mathcal{L} \in IP$
Il existe $\mathcal{P}, \mathcal{V}$ deux machines de Turing qui vérifient la définition 4.
Nous allons créer $\mathcal{M}$ une machine de Turing polynômiale en espace qui reconnaît $\mathcal{L}$.
Comme $PSPACE = NPSPACE$ par le théorème de Savitch, on peut construire $\mathcal{M}$ non déterministe sans invalider la preuve.

On prend $\mathcal{M}$ qui simule $\mathcal{V}$ sur tous les bits aléatoires possibles. Á chaque fois que $\mathcal{V}$ devrait interroger $\mathcal{P}$, on tire parti du non-déterminisme de $\mathcal{M}$ pour "deviner" la réponse qui maximise la probabilité que $\mathcal{V}$ accepte.
On retient le nombre de fois où $\mathcal{V}$ aurait accepté, et le nombre de fois où il aurait refusé, puis on choisit la réponse majoritaire.
Si $\mathcal{M}$ accepte x, alors il existe un $\mathcal{P}$ tel que $\mathbb{P}[< \mathcal{P}, \mathcal{V} > (x) = 1] > \frac{1}{2} > 1 - p$.
De même si $\mathcal{M}$ n'accepte pas x, alors $\forall \mathcal{P}^*, \mathbb{P}[< \mathcal{P}^*, \mathcal{V} > (x) = 1] < \frac{1}{2}$.
Par la définition de IP, on a donc $\mathcal{M}$ accepte $x \iff x \in \mathcal{L}$

Enfin, $\mathcal{M}$ est clairement dans $NPSPACE$ ($\mathcal{V}$ étant en temps polynômial, sa simulation est en espace polynômial).
Donc $IP \subseteq PSPACE$. □

4 $PSPACE \subseteq IP$

Lemme 2. *$PSPACE \subseteq IP$*

Démonstration. On admet que QSAT est un problème $PSPACE$-complet (cf le cours).
Par conséquent, si QSAT est dans IP, alors tous les autres problèmes de $PSPACE$ y sont aussi (car IP est clairement stable par réduction polynômiale).
On va donc trouver une preuve interactive pour QSAT.

Pour ce faire, on transforme les formules booléennes qui sont les instances du problème en formules arithmétiques par la transformation f suivante :

- $f(\forall x_n \phi(x_1, \ldots, x_n)) = f(\phi(x_1, \ldots, x_{n-1}, 0)) \times f(\phi(x_1, \ldots, x_{n-1}, 1))$
- $f(\exists x_n \phi(x_1, \ldots, x_n)) = 1 - (1 - f(\phi(x_1, \ldots, x_{n-1}, 0))) \times (1 - f(\phi(x_1, \ldots, x_{n-1}, 1)))$
- $f(\phi_1 \wedge \ldots \wedge \phi_k) = f(\phi_1) \times \ldots \times f(\phi_k)$
- $f(\phi_1 \vee \ldots \vee \phi_k) = 1 - [(1 - f(\phi_1)) \times \ldots \times (1 - f(\phi_k))]$

On vérifie facilement que $f(\phi) = 1$ si et seulement si ϕ valait vrai, et que sinon $f(\phi) = 0$

Toutefois, un problème de cette transformation est qu'elle produit des polynômes de degré très élevé, ce qui posera problème pour la vérification en temps polynômial de la preuve.
On résout ce problème en remarquant que $\forall k > 0, \forall x_i \in \{0, 1\}, x_i^k = x_i$. On peut donc réduire le polynôme.
En pratique, au lieu de réduire le polynôme final, on fait des réductions après chaque quantificateur (on suppose la formule en forme prénexe), sur toutes les variables non libres, afin de garder le degré du polynôme sous contrôle.

On travaille dans le corps fini à p éléments, p étant un nombre premier choisi par le prouveur. Le vérificateur commence par vérifier que ce nombre est bien premier juste avant le début du protocole proprement dit. Tous les calculs sont faits modulo p, mais p définit avant tout l'intervalle sur lequel le vérificateur peut tirer des nombres aléatoires.
Voici le protocole entre le vérificateur et le prouveur :

- On définit $v_0 = 1$ et $\Phi_0 = f(\phi)$. Dans tout la suite, le prouveur tente de convaincre le vérificateur que $\Phi_k(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n)$ s'évalue en v_k. Seul le vérificateur a accès à v_k.
- Á chaque tour d'interaction, 3 cas se présentent :
  - Cas 1 : Φ_k commence par une multiplication, ce qui correspond à un $\forall$.
    Le prouveur envoie un polynôme de degré 1 $P(x_i)$ au vérificateur (en considérant les autres variables comme fixées, de degré 1 car réduit précédemment).
    Le vérificateur vérifie si $v_k = P(0) \times P(1)$.
    Si non, s'arrêter et renvoyer faux.
    Si oui, tirer r_i un nombre aléatoire plus petit que p (c'est à dire un élément du corps à p éléments) et calculer $v_{k+1} = P(r_i)$.
    Définir Φ_{k+1} comme valant la formule une fois qu'on a remplacé la variable x_i par r_i (et enlevé le quantificateur correspondant).
  - Cas 2 : Φ_k correspond à un $\exists$.
    Ce cas est analogue au précédent (on retire le quantificateur en fixant une variable).

- Cas 3 : Φ_k commence par une réduction, c'est à dire une transformation de tous les x_i^k en x_i pour tout $i < j$ avec un j donné.
  Le prouveur envoie encore un polynôme $P(x_i)$ au vérificateur (mais cette fois pas de degré 1).
  Le vérificateur vérifie si v_k est égale à la réduction de $P(x_i)$ évaluée en r_i. Si non, s'arrêter et renvoyer faux. Si oui, tirer un nouveau r_i (toujours parmi p éléments) et calculer $v_{k+1} = P(r_i)$ Définir Φ_{k+1} comme étant la formule avant que la réduction ne soit faite.
- Lorsque tous les quantificateurs (et les réductions correspondantes) ont disparues, on n'a plus qu'une formule arithmétique sans variables, polynômiale en la taille de l'entrée que le vérificateur peut calculer.

Il est assez clair que dans tout ce protocole, le vérificateur n'accomplit que des actions réalisables en temps polynômial.

Supposons que le vérificateur ait refusé, et que le prouveur était honnête. Alors c'est que la formule initiale était fausse. En effet, on peut remarquer que $v_k = \Phi_k(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n)$ est un invariant, et que le vérificateur ne refuse que s'il est faut à un certain moment.
Si au contraire le prouveur est malhonnête (et que l'énoncé initial est faux), il doit forcément envoyer à un moment un polynôme autre que celui demandé. Toutefois ce polynôme doit coïncider en r_i avec la valeur attendue. Or le prouveur ne peut pas prédire r_i à l'avance et le polynôme qu'il envoie ne peut pas être constant. Il ne peut donc coïncider qu'en au plus autant de point que son degré. La probabilité qu'à l'étape i le prouveur parvienne à "tricher" est donc égale au degré du polynôme qu'il envoie divisé par p.
Le nombre d'étape du protocole est quadratique en la taille de la formule. La probabilité d'erreur du protocole est donc en $\mathcal{O}(n^2 \times d/p)$ où d est le degré maximal des polynômes envoyés par le prouveur. Par ailleurs, $d < n$ car le degré est au plus n au début et que les réductions l'empêchent de dépasser 2 par la suite.
Il suffit donc de choisir p assez grand (ie en $\mathcal{O}(n^4)$) pour avoir un système de preuve interactif de $QSAT$
Donc on a bien $QSAT \in IP$ et $IP \subseteq PSPACE$ □

5 Conclusion

Ce résultat est assez surprenant car non seulement il relie des classes de complexités traditionnelles et des classes de complexités randomisés, mais aussi car il prouve que la randomisation (liée à l'interaction) augmente beaucoup la puissance d'une classe de complexité puisque $\mathcal{V}$ est dans P, à l'aléatoire près.

Plusieurs variantes sont possibles.
Par exemple, il est possible d'avoir une définition asymétrique de IP exigeant que le vérificateur n'ait que des faux positifs et jamais de faux négatifs. Cette preuve fournit un tel vérificateur, donc ces deux définitions sont équivalentes.
Il est aussi possible d'exiger que le prouveur ait ou non accès aux tirages aléatoires du vérificateur. Là encore, puisque cette preuve n'utilise pas le fait que ces tirages soient cachés, ces variantes sont totalements équivalentes (au moins en terme de calculabilité, certains problème peuvent être résolus en moins de round d'interactions si les tirages sont cachés).
Enfin, il est possible de prendre à la fin une borne plus fine en traitant les différents opérateurs différemment (et tout particulièrement en isolant les dernières réductions). Toutefois ceci n'apporte rien en terme de complexité asymptotique, cela permet simplement de spécifier la constante.

Références

[1] Jonathan Katz. *Notes on complexity theory*, 10, nov 2005.

[2] Jonathan Katz. *Notes on complexity theory*, 11, nov 2005.

[3] A. Shen. Ip = pspace : Simplified proof. *J. ACM*, 39, may 1992.