IP = PSPACE

Jean-Bastien Grill

22 décembre 2010

Table des matières

1 La classe IP

1.1 Système de preuve interactive [1]

Définition 1. Un système de preuve interactive est un couple de machines de turing ($\mathcal{P}$,$\mathcal{V}$) tel que, en notant x une entrée et $\mathbb{L}$ un langage :

- $\mathcal{P}$, le prouveur associe à x une preuve de l'assertion : $x \in \mathbb{L}$ (éventuellement incorrecte). Par ailleurs, il dispose d'une quantité de mémoire et d'un temps de calcul infini et a accès à toute l'information sauf aux bits aléatoires éventuellement utilisés par $\mathcal{V}$.
- $\mathcal{V}$, le vérificateur, dialogue avec $\mathcal{P}$ et cherche à vérifier que la preuve donnée par $\mathcal{P}$ est correcte.
  De plus, on note $\mathcal{P}(x)$ la preuve donnée par $\mathcal{P}$ sur l'entrée x.

Remarque 2. *La classe NP peut être décrite par un système de preuve interactive.* $\mathbb{L}$ *est dans NP si* $\exists\mathcal{V}$ *tel que :*

- $\forall x$, $\mathcal{V}$ *dispose d'un temps de calcul polynomial en la taille de* x
- $\forall x$, $\mathcal{V}$ *pose un nombre polynomial de questions en la taille de* x
- $\exists\mathcal{P}\ \forall x$, ($x \in \mathbb{L} \Longrightarrow \mathcal{V}$ *détermine que* $\mathcal{P}(x)$ *est correcte*).
- $\forall\mathcal{P}\ \forall x$, ($x \notin \mathbb{L} \Longrightarrow \mathcal{V}$ *détermine que* $\mathcal{P}(x)$ *est incorrecte*).

1.2 IP

Définition 3. La classe IP peut être décrite par un système de preuve interactive. $\mathbb{L}$ est dans IP si $\exists\mathcal{V}$ tel que :

- $\forall$x, $\mathcal{V}$ dispose d'un temps de calcul polynomial en la taille de x
- $\forall$x, $\mathcal{V}$ pose un nombre polynomial de questions en la taille de x
- $\exists\mathcal{P}\ \forall$x, $\mathbb{P}$($\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte | $x \in \mathbb{L}$) = 1.
- $\forall\mathcal{P}\ \forall$x, $\mathbb{P}$($\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte | $x \notin \mathbb{L}$) $< \frac{1}{2}$.

Informellement, le prouveur cherche à convaincre le vérificateur que $x \in \mathbb{L}$. Si la preuve donnée par le prouveur est correcte le vérificateur l'accepte toujours. Si elle est incorrecte, même le plus astucieux des prouveurs ne pourra pas faire accepter sa preuve au vérificateur avec une probabilité supérieure à $\frac{1}{2}$.

Proposition 4. *NP* $\subseteq$ *IP*

Démonstration. NP est en fait un cas particulier de IP où :

$$\forall\mathcal{P}\ \forall x,\ \mathbb{P}(\ \mathcal{V} \text{ détermine que } \mathcal{P}(x) \text{ est correcte} \mid x \notin \mathbb{L}\ ) = 0 < \frac{1}{2}.$$

□

1. Interactive Proof

2 #3SAT ∈ IP

2.1 Le problème #3SAT

Définition 5. Soit B une formule logique des variables x_1,...,x_n avec n ∈ ℕ. Si B est de la forme :

$$B = \bigwedge_{i=1}^{m} (v_{1,i} \vee v_{2,i} \vee v_{3,i}) \tag{1}$$

Avec $v_{i,j}$ des littéraux de la forme x_k ou $\neg\, x_k$
Alors, en posant #B le nombre d'affectations de variables rendant B vrai et s ∈ ℕ, #3SAT consiste à déterminer si #B = s avec B et s pour entrée.

Proposition 6. *#3SAT ∈ IP*

Pour démontrer cette proposition, on aura d'abord besoin de deux lemmes

2.2 Résultats préliminaires

2.2.1 Conversion d'une formule booléenne en polynôme [2]

Informellement, on cherche à construire à partir d'une formule booléenne un polynôme associé [3] qui, si l'on change chaque "vrai" en 1 et chaque "faux" en 0, se comporte exactement comme cette formule booléenne. L'idée est la suivante : trouver le nombre d'affectations rendant une formule vraie se ramène à chercher le nombre de fois que le polynôme associé vaut 1.

Définition 7. Soit n ∈ ℕ et B une formule booléenne des variables x_1,...x_n. Puisque a ∨ b = ¬(¬a ∧ ¬b), on peut supposer que B n'est formé que de ∧, ¬ et des variables x_1,...,x_n.
On définit alors le polynôme P_B associé à B par induction :
Si C et D sont des formules booléennes des variables x_1,..,x_n, on pose respectivement P_C et P_D leur polynôme associé. On définit alors P_B par :

$$P(X_1, ..., X_n) = \begin{cases} P_C(X_1, ..., X_n).P_D(X_1, ..., X_n) & \text{si } B = C \wedge D \\ 1 - P_C(X_1, ..., X_n) & \text{si } B = \neg C \\ X_i & \text{si } B = x_i \end{cases} \tag{2}$$

Définition 8. Soit a ∈ {vrai,faux}, on pose [4] δ : {vrai,faux} → {0,1} défini par :

$$\delta(a) = \begin{cases} 1 & \text{si } a = vrai \\ 0 & \text{si } a = faux \end{cases} \tag{3}$$

Exemple 9.

$$B = x_1 \wedge (\neg x_1 \vee x_2) \tag{4}$$

$$P_B = X_1(1 - X_1(1 - X_2)) \tag{5}$$

2. cette conversion est aussi appelée arithmétisation, Arithmetization en anglais
3. L'appellation associée n'est pas standardisée, elle n'est adoptée que pour ce document
4. Cette notation n'est pas standardisée

Exemple 10.

$$B = \bigwedge_{i=1}^{m} (v_{1,i} \vee v_{2,i} \vee v_{3,i}) \tag{6}$$

On peut alors expliciter P_B

$$P_B = \prod_{i=1}^{m} (1 - (1 - P_{v_{1,i}})(1 - P_{v_{2,i}})(1 - P_{v_{3,i}})) \tag{7}$$

Les $v_{j,i}$ étant soit des x_k soit des $\neg\, x_k$, les $P_{v_{j,i}}$ sont de degré 1 et donc P_B est de degré 3m.

Lemme 11. *Soit $n \in \mathbb{N}$, B une formule logique de n variables et P_B son polynome associé. Alors, $\forall\, x_1,\ldots,x_n \in$ {vrai, faux} :*

$$B(x_1,\ldots x_n) = vrai \iff P_B(\delta(x_1),\ldots\delta(x_1)) = 1$$

Démonstration. On procède par induction :

(i) Si B = C $\wedge$ D alors on vérifie que :

$$\begin{aligned} B = vrai &\iff \text{C = vrai et D = vrai (par définition de } \wedge) \\ &\iff P_C = 1 \text{ et } P_D = 1 \text{ (par hypothèse d'induction)} \\ &\iff P_C.P_D = 1 \text{ (car si } P_C \text{ ou } P_D \text{ est nul alors le produit est nul)} \end{aligned} \tag{8}$$

(ii) Si B = $\neg$ C alors on vérifie que :

$P_C = 1 \Longrightarrow P_B = 1 - P_C = 1 - 1 = 0$

$P_C = 0 \Longrightarrow P_B = 1 - P_C = 1 - 0 = 1$

□

Ce lemme nous assure que le polynôme associé ainsi construit a bien la propriété que l'on désirait, c'est-à-dire de se comporter exactement comme sa formule logique. On peut donc indifféremment travailler sur la formule booléenne ou sur son polynôme associé.

2.2.2 Tester efficacement l'égalité de deux polynômes de façon probabiliste

On cherche un algorithme polynomial testant l'égalité de deux polynômes à une variable de façon probabiliste. C'est à dire un algorithme Π tel que : $\exists$ p $<$ 1, $\forall$ P,Q polynômes,

$$\mathbb{P}(\Pi \text{ renvoie P} \neq \text{Q} \mid \text{P} = \text{Q}) = 0 \tag{9}$$

$$\mathbb{P}(\Pi \text{ renvoie P} = \text{Q} \mid \text{P} \neq \text{Q}) < p \tag{10}$$

Π ne se trompe jamais si P = Q mais peut se tromper si P $\neq$ Q.

Algorithme 1 détermine si P = Q

Require: l'évaluation de P et Q en un point soit polynomiale en leur degré

1: $u \leftarrow$ un élément choisi uniformément aléatoirement dans un ensemble fini $\mathbb{S}$
2: **if** P(u) = Q(u) **then**
3: **return true**
4: **else**
5: **return false**

Lemme 12. *On note d = max(degré(P),degré(Q)). L'algorithme 1 noté* Π *est polynomial en d et vérifie :*

$$\mathbb{P}(\Pi \text{ renvoi } P \neq Q \mid P = Q) = 0 \tag{11}$$

$$\mathbb{P}(\Pi \text{ renvoi } P = Q \mid P \neq Q) \leq \frac{d}{Card(\mathbb{S})} \tag{12}$$

Démonstration.

(i) L'instruction ligne 1 est effectué en temps constant. L'instruction ligne 2 est effectué en temps polynomial en d par hypothèse.
Le temps de calcul est donc bien polynomial en d.

(ii) Si P = Q, il est clair que : $\forall$u, P(u) = Q(u) et donc :
$$\mathbb{P}(\Pi \text{ renvoi P} \neq \text{Q} \mid \text{P} = \text{Q}) = 0$$
Supposons maintenant que P $\neq$ Q.
- D'une part, P - Q est non nul car P $\neq$ Q.
- D'autre part, P - Q est de degré au plus d car d = max(degré(P),degré(Q)).
P - Q a donc au plus d racines. Ainsi la probabilité que u choisi uniformément dans $\mathbb{S}$ soit racine de P - Q, est majorée par $\frac{d}{Card(\mathbb{S})}$

□

Remarque 13. *L'algorithme déterministe qui consiste à comparer chaque coefficient s'exécute aussi linéairement en le degré. Néanmoins, l'algorithme présenté ci-dessus est tout de même intéressant car il ne nécessite pas une explicitation des polynômes mais seulement une manière de les calculer en un point en temps polynomial.*

2.3 #3SAT $\in$ IP et conséquences

Nous allons maintenant utiliser les deux idées ci-dessus pour démontrer notre affirmation de départ. Nous la rappelons :

Proposition 14. *#3SAT* $\in$ *IP*

Démonstration. Soit B une conjonction de m clauses, m $\in \mathbb{N}$. On pose $P_B(X_1,...X_n)$ le polynôme associé à B. Dans l'exemple 10, nous avons remarqué que P_B est

de degré 3m.
Pour i $\in$ {0,...,n}, on pose :

$$F_i(x_1, ..., x_i) = \sum_{x_{i+1},...,x_n \in \{0,1\}} P_B(x_1, ...x_n) \tag{13}$$

F_i(x_1,...,x_i) représente le nombre d'affectations rendant B vrai aux variables x_1,...,x_i fixées.
On cherche : F_0 = #B. D'abord, quelques propriétés des fonctions F_i :

$$F_0 = \#P_B \tag{14}$$

$$F_n(x_1, ..., x_n) = P_B(x_1, ..., x_n) \tag{15}$$

$$F_{i-1}(x_1, ..., x_{i-1}) = F_i(x_1, ..., x_{i-1}, 0) + F_i(x_1, ..., x_{i-1}, 1) \tag{16}$$

Nous allons maintenant construire $\mathcal{V}$.
$\mathcal{V}$ commence par obtenir un nombre premier p dans]6n.2^n,6n.2^{n+1}]. C'est possible car vérifier qu'un nombre est premier, est polynomial en sa taille. De plus on peut démontrer qu'un tel entier existe quelque soit n.
On voit alors les coefficients des F_i dans $\mathbb{Z}/p\mathbb{Z}$. Ceci ne pose pas de problème car #B $\leq 2^n$.
$\mathcal{V}$ demande F_1 à $\mathcal{P}$. Notons g la réponse de $\mathcal{P}$. On sait que #B = F_0 = F_1(0) + F_1(1).
Ainsi, si g(0) + g(1) $\neq$ s alors $\mathcal{V}$ peut répondre que #B $\neq$ s.
Sinon, il faut encore vérifier que g = F_1. Pour cela on utilise l'algorithme vu ci-dessus qui teste l'égalité de deux polynômes.
Tout d'abord $\mathcal{V}$ génère un nombre u uniformément dans $\mathbb{Z}/p\mathbb{Z}$ et cherche à savoir si : g(u) = F_1(u).
On pose ensuite P'(X_2,...,X_n) = P_B(u,X_2,...,X_n). Déterminer si g(u) = F_1(u) revient à chercher si g(u) = SOMME P'. Ce problème est exactement le problème initial avec g(r) qui prend le rôle de s et P' le rôle de P_B et peut donc être résolu récursivement.
Le cas de base survient lorsque le polynôme ne dépend d'aucune variable et est donc constant.

Chacune des étapes de l'algorithme est polynomial. En effet, il suffit de vérifier que :

- Obtenir le nombre premier p est polynomial en n car vérifier la primalité est polynomial.
- Calculer la somme g(0) + g(1) est polynomial en n (car logarithmique en p)
- Choisir u uniformément dans $\mathbb{Z}/p\mathbb{Z}$ est polynomial en n (car logarithmique en p)
- A chaque appel récursif, le nombre de variables diminue de 1. Il y a donc exactement n appels récursifs, c'est-à-dire un nombre polynomial en n d'appels récursifs.

Enfin on vérifie que :

$$\exists \mathcal{P} \forall x, \mathbb{P}(\ \mathcal{V} \text{ détermine que } \#\mathrm{B} = \mathrm{s} \mid \#\mathrm{B} = \mathrm{s}\ ) = 1. \tag{17}$$

En effet, si $\mathcal{P}$ donne le bon F_1 à chaque appel récursif et que #B = s, alors $\mathcal{V}$ va renvoyer vrai. Ceci repose sur le fait que si P = Q alors P(u) = Q(u).
On vérifie aussi que :

$$\forall \mathcal{P} \forall x, \mathbb{P}(\ \mathcal{V} \text{ détermine que } \#\mathrm{B} = \mathrm{s} \mid \#\mathrm{B} \neq \mathrm{s}) < \frac{1}{2}. \tag{18}$$

En effet, d'après ce qui précède, on sait que, à chaque appel récursif :
$\mathbb{P}(\ \mathcal{V}$ détermine que g = $F_1 \mid \mathrm{g} \neq F_1) < \frac{3m}{p}$. Car le degré de F_1 est majoré par 3m.
Ainsi, $\forall \mathcal{P}$ $\forall$ x, $\mathbb{P}(\ \mathcal{V}$ détermine que #B = s | #B $\neq$ s) $< \frac{3m.n}{p} < \frac{1}{2}$.

□

Remarque 15. *La démonstration ci-dessus n'utilise pas la forme particulière des polynôme à associé à des disjonctions de 3-clauses mais seulement le fait que ces polynômes soient de degré polynomial en la taille de la formule booléenne (de degré 3m avec m le nombre de clauses).*

Corrolaire 16.

$$NP \subseteq IP \tag{19}$$

$$co - NP \subseteq IP \tag{20}$$

Démonstration.

(i) 3SAT est réductible à #3SAT en cherchant si #B ≥ 1
(ii) co-3SAT est réductible à #3SAT en cherchant si #B = 0

□

Le fait que NP $\subseteq$ IP a déjà été vu en remarque. Cela découle du fait que NP peut être vu comme l'ensemble des problèmes dont la preuve est vérifiable en temps polynomial. Le résultat co- NP $\subseteq$ IP est moins trivial. On a en fait un résultat plus général :

Proposition 17. *IP est clôt par complémentation*

Démonstration. Lorsqu'on aura prouvé que IP = PSPACE, PSPACE étant clôt par complémentation, on aura établi la proposition. □

3 IP = PSPACE

Cette partie se propose de démontrer l'égalité entre la classe des problèmes IP et celle des problèmes PSPACE. Pour celà on montrera d'abord l'inclusion IP $\subseteq$ PSPACE puis on montrera l'inclusion inverse.
La classe IP peut être vue comme une généralisation de la classe NP où on autorise une probabilité d'erreur. On voit que cet ajout augmente la puissance de la classe.
Comme nous l'avons vu précédemment, l'égalité IP = PSPACE permet aussi de montrer facilement des propriétés non triviales sur IP telle que sa clôture par complémentation.

La première inclusion, IP $\subseteq$ PSPACE est facile à démontrer. Pour créer une machine utilisant un espace polynomial à partir d'un vérificateur $\mathcal{V}$ il suffit de générer toutes les réponses possibles des prouveurs.
Pour la deuxième inclusion nous adapterons la démontsration de #3SAT $\in$ IP pour prouver que #SAT (qui est PSPACE complet) est dans IP.

3.1 IP $\subseteq$ PSPACE

Proposition 18. *IP $\subseteq$ PSPACE*

Démonstration. On se donne un langage $\mathbb{L}$ dans IP, on dispose donc d'un vérificateur $\mathcal{V}$ qui vérifie :
- $\forall$x, $\mathcal{V}$ dispose d'un temps de calcul polynomial en la taille de x
- $\forall$x, $\mathcal{V}$ pose un nombre polynomial de questions en la taille de x
- $\exists\mathcal{P}$ $\forall$x, $\mathbb{P}$($\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte | x $\in \mathbb{L}$) = 1.
- $\forall\mathcal{P}$ $\forall$x, $\mathbb{P}$($\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte | x $\notin \mathbb{L}$) $< \frac{1}{2}$.

Puisque $\mathcal{V}$ utilise un nombre polynomial de bits aléatoires, on peut générer en espace polynomial tous les calculs de $\mathcal{V}$ possibles sur une réponse de $\mathcal{P}$ et stocker le nombre de calculs qui aboutissent à l'acceptation de la preuve donnée par $\mathcal{P}$.
On peut donc déterminer en espace polynomial :

$$\mathbb{P}(\mathcal{V} \text{ détermine que } \mathcal{P}(\text{x}) \text{ est correcte})$$

De plus, on peut générer toutes les réponses possibles de $\mathcal{P}$ en espace polynomial puisque les réponses sont de taille polynomiale. On peut alors déterminer :

$$\max_{\mathcal{P}} \mathbb{P}(\mathcal{V} \text{ détermine que } \mathcal{P}(\text{x}) \text{ est correcte})$$

Enfin on remarque que :

$$\max_{\mathcal{P}} \mathbb{P}(\mathcal{V} \text{ détermine que } \mathcal{P}(\text{x}) \text{ est correcte}) = 1 \iff x \in \mathbb{L} \tag{21}$$

Puisque :
- Si x $\in \mathbb{L}$ alors $\exists\mathcal{P}$, $\mathbb{P}(\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte) = 1 et donc $\max_{\mathcal{P}}$ $\mathbb{P}(\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte) = 1.

- Si x $\notin \mathbb{L}$ alors $\forall \mathcal{P}$, $\mathbb{P}(\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte) $< \frac{1}{2}$ et donc $\max_{\mathcal{P}}$ $\mathbb{P}(\mathcal{V}$ détermine que $\mathcal{P}$(x) est correcte) $< \frac{1}{2}$ et est en particulier différent de 1.

□

3.2 PSPACE ⊆ IP

Proposition 19. *PSPACE ⊆ IP*

Lemme 20. *Soit B une formule booléenne quelconque. On définit sa taille comme étant le nombre de caractères (hors parenthèse) utilisés pour l'écrire. On a le résultat suivant :*
Le degré de P_B est polynomial en la taille de B.

Démonstration. On prouve par induction la propriété plus forte suivante :
Le degré de P_B est majoré par la taille de B.

- Si la formule est réduite à une variable, alors le polynôme associé est de degré 1 et la taille de la formule est aussi 1
- Si B = ¬C alors P_B = 1 - P_C d'où deg(P_B) ≤ deg(P_C). De plus taille(B) = taille(C) + 1. Or par hypothèse d'induction : deg(P_C) ≤ taille(C). Ainsi, deg(P_B) ≤ deg(P_C) ≤ taille(C) ≤ taille(B). On a donc bien : deg(P_B) ≤ taille(B)
- Si B = C ∧ D alors : $P_B = P_C.P_D$. On a :

$$\begin{cases} deg(P_B) = deg(P_C) + deg(P_D) \\ taille(B) = taille(C) + taille(D) + 1 \\ deg(P_C) \leq taille(C) \text{ et } deg(P_D) \leq taille(D) \text{ (par hypothèse d'induction)} \end{cases} \tag{22}$$

  D'où deg(P_B) ≤ taille(C) + taille(D) ≤ taille(B)
- Si B = C ∨ D alors : P_B = 1 - (1 - P_C)(1 - P_D) d'où deg(P_B) = deg(P_C) + deg(P_D). On est alors dans la même situation que pour B = C ∧ D et comme expliqué ci-dessus : deg(P_B) ≤ taille(B).

□

Démonstration. On peut maintenant démontrer la proposition.
Notons #SAT le problème qui consiste à déterminer le nombre d'affectations rendant une formule booléenne quelconque vraie.
On se propose de montrer que #SAT ∈ IP ce qui prouverait l'inclusion car on sait par ailleurs que #SAT est PSPACE complet.

Montrer #SAT ∈ IP n'est pas plus difficile que #3SAT ∈ IP. En effet, on avait remarqué que la démonstration de #3SAT ∈ IP n'utilisait pas la forme particulière du polynôme associé mais juste que son degré était polynomial en la taille de la formule.
Or on a justement montré en lemme que le degré du polynôme associé à une formule booléenne était polynomial en sa taille.

La fin de la démonstration est donc exactement identique à la démonstration de #3SAT $\in$ IP. $\square$

Références

[1] R. Motwani and P. Raghavan, Randomized Algorithms. Cambridge University Press, 1995.

[2] O. Carton, Langages formels, calculabilité et complexité, 2008

[3] Papadimitriou, C. H. Computational Complexity, 1994