Théorème de Gill-Solovay-Baker

N.Blanchard

Machines à Oracle

Premières définitions

Extensions aux classes de complexité

Énumération des machines

Preuve du théorème

Énoncé

$P^{\mathcal{A}} = NP^{\mathcal{A}}$

$P^{\mathcal{B}} \neq NP^{\mathcal{B}}$

Conclusion

Théorème de Gill-Solovay-Baker

N.Blanchard

Département d'Informatique
École Normale Supérieure

20 décembre 2013

Machines à Oracle
Premières définitions
Extensions aux classes de complexité
Énumération des machines
Preuve du théorème
Énoncé
$P^{\mathscr{A}} = NP^{\mathscr{A}}$
$P^{\mathscr{B}} \neq NP^{\mathscr{B}}$
Conclusion

Plan de la présentation

1. Machines à Oracle
   - Premières définitions
   - Extensions aux classes de complexité
   - Énumération des machines
2. Preuve du théorème
   - Énoncé
   - $\mathbf{P}^{\mathscr{A}} = \mathbf{NP}^{\mathscr{A}}$
   - $\mathbf{P}^{\mathscr{B}} \neq \mathbf{NP}^{\mathscr{B}}$
3. Conclusion

Oracles

- Boîte noire qui permet de décider si un mot est dans un langage $\mathscr{A}$
- Modèle classique plus une bande supplémentaire, et trois états :
  - ASK
  - YES
  - NO
- En temps 1 on passe de l'état ASK à l'état réponse selon que le mot soit ou non dans $\mathscr{A}$

Machines à Oracle

- Effectue un calcul normal en passant un nombre quelconque de fois par l'état ASK
- On se restreindra ici à $\mathscr{A}$ décidable mais ce n'est pas nécessaire
- $M^{\mathscr{A}}$ est la machine M munie d'un oracle décidant $\mathscr{A}$

Classes à Oracles

- On peut considérer toutes les machines de $\mathbf{C}$ ayant accès à un oracle $\mathscr{A}$:

$$\mathbf{C}^{\mathscr{A}} = \bigcup_{M \in \mathbf{C}} M^{\mathscr{A}}$$

- On peut aussi considérer les machines ayant accès à une classe d'oracle $\mathbf{B}$:

$$\mathbf{C}^{\mathbf{B}} = \bigcup_{\mathscr{A} \in \mathbf{B}} \mathbf{C}^{\mathscr{A}}$$

- Si $\mathscr{A}$ est $\mathbf{B}-\mathbf{Complet}$, on a (généralement) $\mathbf{C}^{\mathbf{B}} = \mathbf{C}^{\mathscr{A}}$

Techniques Relativisantes

- Prenons une preuve de $\mathbf{C} = \mathbf{B}$
- La preuve est relativisante si elle implique que pour tout $\mathscr{A}$, $\mathbf{C}^{\mathscr{A}} = \mathbf{B}^{\mathscr{A}}$
- De nombreuses techniques en CC sont relativisantes (e.g. preuve du théorème de hiérarchie en temps)

Énumération

- On peut énumérer les langages décidables
- On peut aussi coder les couples (machine, oracle) quand l'oracle est décidable
- On utilisera par la suite une énumération des machines à oracle sans préciser l'oracle :
  M_i est la i-ème machine à oracle

Théorème de Gill-Solovay-Baker

Théorème

Il existe un langage $\mathscr{A}$ *tel que* $\mathbf{P}^{\mathscr{A}} = \mathbf{NP}^{\mathscr{A}}$

Théorème

Il existe un langage $\mathscr{B}$ *tel que* $\mathbf{P}^{\mathscr{B}} \neq \mathbf{NP}^{\mathscr{B}}$

Corollaire

Aucune technique relativisante ne peut résoudre **P vs NP**

Langage **TQBF**

- Langage des formules booléennes quantifiées
- Appartient **PSPACE** car il suffit de tester chacune des possibilités pour les variables
- Est **PSPACE – Complet**, preuve de Meyer-Stockmeyer :
  - Soit $\mathscr{M}$ une machine calculant un langage de **PSPACE** en espace $p(m)$, on peut calculer une formule de longueur polynomiale $n = p'(m)$ décrivant une configuration de $\mathscr{M}$
  - Il suffit de montrer qu'il existe un chemin dans le graphe de configuration allant du départ A à un état acceptant B de longueur au plus 2^n, que l'on note $F_n(A, B)$

PSPACE – Completude

- La preuve s'inspire de celle du théorème de Savitch, qu'on donne en schéma :
  - Dire $F_n(A, B)$ revient à dire qu'il existe un C et deux chemins de longueur 2^{n-1}entre (A, C) et (C, B)
  - On a une forme récursive, qui transforme naturellement $F_n(A, B)$ en $\exists C, F_{n-1}(A, C) \wedge F_{n-1}(C, B)$, ce qui ne marche pas de manière naïve
  - On prend à la place la formule suivante

    $$\exists C \forall X \forall Y ((X = A \wedge Y = C) \vee (X = C \wedge Y = B)) \Rightarrow F_{n-1}(X, Y)$$

  - La formule finale est de taille polynomiale ce qui achève la preuve du théorème

$\mathbf{P}^{\mathbf{TQBF}} = \mathbf{NP}^{\mathbf{TQBF}}$

- On peut montrer les inclusions suivantes :
  - $\mathbf{P}^{\mathbf{TQBF}} \subseteq \mathbf{NP}^{\mathbf{TQBF}}$
  - $\mathbf{NP}^{\mathbf{TQBF}} \subseteq \mathbf{NPSPACE}^{\mathbf{TQBF}} = \mathbf{NPSPACE}$
  - $\mathbf{NPSPACE} = \mathbf{PSPACE}$ (Immerman–Szelepcsényi)
  - $\mathbf{PSPACE} \subseteq \mathbf{P}^{\mathbf{TQBF}}$
- On a donc bien un langage $\mathscr{A}$ tel que $\mathbf{P}^{\mathscr{A}} = \mathbf{NP}^{\mathscr{A}}$
- D'autres langages auraient aussi marché (des langages complets plus puissants que **TQBF** par exemple)

Définition de $\mathscr{B}$

- On commence avec $\mathscr{B} = \emptyset$, et on définit $\mathscr{B}$ récursivement
- À l'étape i, supposons que le mot le plus long ajouté jusqu'ici était de longueur $n-1$
  On lance la machine $\mathscr{M}_i$ sur l'entrée 1^n pendant $\frac{2^n}{10}$ étapes.
  Quand elle fait appel à l'oracle sur un mot q :
  - Si on a déjà déterminé si q était dans $\mathscr{B}$ on répond de manière cohérente
  - Sinon on répond NON
- Si la machine termine en temps inférieur à $\frac{2^n}{10}$ on agit ainsi :
  - Si elle accepte on définit $\mathscr{B} \cap \{0,1\}^n = \emptyset$
  - Sinon on ajoute à $\mathscr{B}$ un mot de longueur n dont l'appartenance n'est pas encore déterminée

Argument Diagonal

- On considère désormais le langage $\mathscr{U}_{\mathscr{B}}$, tel que $1^n \in \mathscr{U}_{\mathscr{B}} \Leftrightarrow \exists w \in \{0,1\}^n \cap \mathscr{B}$
- Si M_i s'arrête sur 1^n, alors
  - Ou bien elle accepte et on sait que $\mathscr{B} \cap \{0,1\}^n = \emptyset$
  - Ou bien elle refuse et il existe un mot de longueur n dans $\mathscr{B}$
- Dans les deux cas M_i se trompe si elle s'arrête en temps inférieur à $\frac{2^n}{10}$
- Il n'existe donc aucune machine déterministe à oracle décidant le langage $\mathscr{U}_{\mathscr{B}}$ en temps polynomial.

$\mathbf{P}^{\mathscr{B}} \neq \mathbf{NP}^{\mathscr{B}}$

- On a vu que $\mathscr{U}_{\mathscr{B}}$ n'appartenait pas à $\mathbf{P}^{\mathscr{B}}$.
- Si il existe un mot de longueur n dans $\mathscr{B}$, on peut le deviner de manière non déterministe.
- En utilisant un oracle sur $\mathscr{B}$, on peut donc deviner un mot de longueur n et vérifier qu'il est bien dans $\mathscr{B}$ avec l'oracle.
- On a donc bien $\mathscr{U}_{\mathscr{B}} \in \mathbf{NP}^{\mathscr{B}}$ et donc $\mathbf{P}^{\mathscr{B}} \neq \mathbf{NP}^{\mathscr{B}}$

Extensions à **P vs NP**

- Il existe plus de 100 preuves annoncées (fausses !) de **P vs NP** (dont plus de 15 publiées)
- Si on a une preuve relativisante de $\mathbf{P} = \mathbf{NP}$, on a aussi une preuve de $\mathbf{P}^{\mathscr{B}} = \mathbf{NP}^{\mathscr{B}}$
- De même pour les preuves de $\mathbf{P} \neq \mathbf{NP}$
- On a donc un méta-théorème qui permet d'écarter toutes les preuves relativisantes directement.

Quelques références

1. **Computational Complexity, A Modern Approach**, de Sanjeev Arora et Boaz Barak
2. Le cours de complexité de Jean Goubault-Larrecq
3. Le site de Gerhard J. Woeginger pour une liste de fausses preuves de **P vs NP**