Théorème de Ladner

Weikun HE

7 février 2011

1 Énoncé du théorème

Dans la classe **NP**, on connaît beaucoup de langages décidables en temps polynomial et un très grand nombre de problèmes naturels sont démontrés **NP**-complet. Une question naturelle à se poser est : est-ce qu'il existe un langage dans **NP** qui n'est ni **P** ni **NP**-complet ? Malheureusement, on ne sait pas encore répondre à cette question. Mais on sait que si l'on suppose $\mathbf{P} \neq \mathbf{NP}$ alors la réponse est OUI. C'est ce que montre le théorème suivant.

Théorème 1 (R. E. Ladner, 1975)**.** *Si* $\mathbf{P} \neq \mathbf{NP}$*, alors il existe des langages qui ne sont ni* **P** *ni* **NP***-complets.*

2 Démonstration

Le théorème se déduit immédiatement du lemme suivant.

Lemme 2. *Si* $L \in \mathbf{NP} \setminus \mathbf{P}$*, alors il existe un langage* $L' \in \mathbf{NP} \setminus \mathbf{P}$ *tel que* L' *se réduit (en temps polynomial) à* L *mais* L *ne se réduit pas à* L'*.*

En fait, ce lemme est plus fort que le théorème et a des conséquences intéressantes.

Démonstration. Notons M_i la machine de Turing codée par i.

Soit L un langage dans $\mathbf{NP} \setminus \mathbf{P}$. On définit un langage L' et une fonction $f : \mathbb{N} \to \mathbb{N}$ de manière récursive. On pose $L' = \{w01^{n^{f(n)}} \mid w \in L \text{ et } n = |w|\}$. Les mots de L' sont donc les mots w de L suivis d'un 0 et d'une suite de 1 d'une certaine longueur précise.

Et pour $n \in \mathbb{N}$, $f(n)$ est, s'il existe, le plus petit $i < \log\log n$ tel que pour toute entrée $w \in \{0,1\}^{\leq \log\log n}$, M_i accepte w au bout de $|w|^i$ étapes de calcul si et seulement si $w \in L'$ (i.e. la machine M_i s'arrête en temps $|w|^i$ et son langage associé coïncide avec L' sur les mots de longueur inférieure à $\log\log n$. Et pour simuler on peut utiliser une machine de Turing universelle munie d'un « horloge ».) Et si un tel i n'existe pas, on pose $f(n) = \log\log n$.

Donnons quelques propriétés de f :

Lemme 3. *On a*

1. *Le calcul de $f(n)$, comme tel qu'on l'a décrit, s'arrête en un temps polynomial en n.*
2. *f est croissante.*
3. *f est stationnaire si $L' \in \mathbf{P}$.*
4. *f tend vers l'infini si $L' \notin \mathbf{P}$.*

Démonstration. 1. Notons $T_1(n)$ le temps de calcul de $f(n)$ et $T_2(m)$ le temps pour décider l'appartenance à L' d'un mot de longueur au plus m. Pour calculer $f(n)$, on simule au plus $\log\log n$ machines de Turing, chaque machine sur au plus $2^{\log\log n} = \log n$ entrée, et chaque entrée pour au plus $(\log\log n)^{\log\log n}$ étapes. Ensuite, on lance la machine acceptant L' pour tous les mots de longueur inférieure à $\log\log n$. Donc

$$T_1 \leq O(n) + \log n T_2(\log\log n)$$

Pour décider si un mot de longueur de m appartient à L'. On a besoin d'une machine qui accepte L. Or $\mathbf{NP} \subset \mathbf{EXPTIME}$, on peut donc supposer que cette machine tourne en temps $2^{O(m^c)}$ donc

$$T_2(m) \leq 2^{O(m^c)} + T_1(m)$$

Les deux inégalités entraînent

$$T_1(n) \leq O(n) + \log n 2^{O((\log\log n)^c)} + \log n T_1(\log\log n)$$

et comme $2^{O((\log\log n)^c)} = 2^{O(\log n)} = n^{O(1)}$, on a

$$T_1(n) \leq n^{O(1)} + \log n T_1(\log\log n)$$

On en déduit que T_1 est polynomial.

2. $\forall n$, on a $\forall i < f(n)$, la machine M_i s'arrêtant en $|w|^i$ étapes ne décide pas l'appartenance à L' pour les mots de longueur $< \log\log n$ et $i \leq \log\log(n+1)$ donc $i \neq f(n+1)$. Par suite, $f(n) \leq f(n+1)$.
3. Si $L' \in \mathbf{P}$ alors, il existe une machine M qui décide $w \in L'$ en temps $|w|^c$. Or chaque machine est codée par une infinité de codes, (car on peut ajouter, en outre, des états inutiles), donc $\exists i \geq c$ tel que $M = M_i$ donc chaque fois qu'on simule M_i, elle donne la bonne réponse.
Donc $\forall n, f(n) \leq i$. Or une suite croissante bornée à valeur entière est stationnaire donc f est stationnaire.
4. Supposons que f ne diverge pas vers l'infini, comme elle est croissante, elle est dans ce cas stationnaire en un nombre i. Alors la machine M_i décide le langage L' car $\forall w \in \{0,1\}^*$ comme la suite $\log\log n$ tend vers l'infini, $\exists n$ tel que $|w| < \log\log n$ et $f(n) = i$. Par la définition f, M_i accepte w si et seulement si $w \in L'$. D'où la contraposée de ce qu'il fallait démontrer.

□

Remarquons que l'on a d'une certaine façon énuméré toutes les machines de Turing en temps polynomial comme on le voit dans la démonstration des points 3 et 4 du lemme précédent.

Maintenant montrons que L' a bien des propriétés souhaitées.

Lemme 4. *$L' \leqslant_p L$ et par conséquent, $L' \in \mathbf{NP}$.*

Démonstration. Fixons un mot v_0 qui n'est pas dans L, ceci est possible parce que $L \notin \mathbf{P}$. Décrivons la fonction qui réduit L' à L.

Entrée: w
  Si w ne contient pas de 0 Alors
    rejeter
  Sinon
    récupérer l'unique mot v et l'unique nombre k tel que $w = v01^k$ {*cela se fait en temps linéaire*}
    Si $k = |v|^{f(|v|)}${*on a vu que cela prend un temps polynomial*} Alors
      retourner v
    Sinon
      retourner v_0

C'est bien une réduction car ce n'est rien d'autre que la définition de L'. Donc $L' \leqslant_p L$, et comme $L \in \mathbf{NP}$, on a $L' \in \mathbf{NP}$. □

Lemme 5. *$L' \notin \mathbf{P}$.*

Démonstration. Supposons que $L' \in \mathbf{P}$. Alors d'après le lemme 3.3, $\exists i, \exists n_0, \forall n \geq n_0, f(n) = i$. La fonction suivante réduirait L à L'.

Entrée: w
  Si $n = |w| < n_0$ Alors
    calculer $f(n)$
    retourner $w01^{n^{f(n)}}$
  Sinon
    retourner $w01^{n^i}$

Cela se fait en temps $O(n^i)$. Donc $L \leqslant_p L'$, par suite, L serait dans $\mathbf{P}$ ce qui contredit notre hypothèse. □

Lemme 6. *$L \not\leqslant_p L'$*

Démonstration. D'après lemme précédent, $L' \notin \mathbf{P}$. Donc, d'après le lemme 3.4, f diverge vers l'infini.
Raisonnons par absurde. On suppose qu'il existe une fonction $g : \{0,1\}^* \to \{0,1\}^*$ calculable en temps n^c telle que g réduit L à L'. Donc un mot w est dans L si et seulement si $g(w)$ est de la forme $g(w) = v01^k$ avec $k = |v|^{f(|v|)}$.
Comme f tend vers l'infini, il existe un rang n_0 tel que $\forall n \geq n_0$, $f(n) \geq 2c$.
L'idée est de combiner cette réduction g et la réduction $L' \leqslant_p L$ pour obtenir un algorithme récursif en temps polynomial. Voici un algorithme A qui décide le langage L.

Entrée: w
  Si $g(w)$ ne contient pas de 0 Alors
    rejeter
  Sinon
    trouver v et k tels que $g(w) = v01^k$
    Si $k \neq |v|^{f(|v|)}$ Alors
      rejeter
    Sinon
      Si $|v| < n_0$ Alors
        déterminer si $v \in L \cap \{0,1\}^{\leq n_0}$ {*en temps constant puisque le langage est fini*}
      Sinon
        retourner $A(v)$ {*c'est un appel récursif*}

Il s'agit d'un algorithme récursif en temps polynomial. En effet, pour l'entrée w, dans le cas où la récursion a lieu, $g(w)$ est de la forme $g(w) = v01^k$ avec $k = |v|^{f(|v|)}$ et $|v| \geq n_0$. Donc $|g(w)| = |v| + 1 + k$. Or $|g(w)|$ est bornée par le temps de calcul donc $|g(w)| \leq |w|^c$. Il s'en suit que $|v|^{f(|v|)} \leq |w|^c$ et $|v| \leq w^{1/2}$ car $f(|v|) \geq 2c$. Le temps de calcul vérifie donc $T(n) \leq n^{O(1)} + T(n^{1/2})$. On en déduit que T est polynomial.
Donc on a une machine qui décide L en temps polynomial, ce qui contredit encore une fois l'hypothèse $L \notin \mathbf{P}$ □

On conclut que L' est dans $\mathbf{NP} \setminus \mathbf{P}$ et que L' se réduit à L mais L ne se réduit pas à L'. Cela signifie que L' est strictement plus « simple » que L. En particulier L' n'est pas $\mathbf{NP}$-complet. □

3 Conséquences

Donc si $\mathbf{P} \neq \mathbf{NP}$ alors il existe des langages dans $\mathbf{NP} \setminus \mathbf{P}$ non $\mathbf{NP}$-complets. Ces langages forment une classe que l'on appelle $\mathbf{NP}$-*intermédiaire* et que l'on note $\mathbf{NPI}$. Bien sûr, on ne sait pas encore si cette classe existe réellement. Si on regarde de plus près la classe $\mathbf{NPI}$, le lemme 2 nous donne des informations plus précises.

Corollaire 7. *Si $\mathbf{P} \neq \mathbf{NP}$ alors il y a une infinité de hiérarchies de langages dans $\mathbf{NPI}$.*

La structure dans $\mathbf{NPI}$ est donc très compliquée.

Démonstration. Il suffit d'appliquer le lemme 2 pour obtenir une suite de langages $L_1, L_2, L_3, \ldots$ dans $\mathbf{NP} \setminus \mathbf{P}$ telle que $L_{i+1} \leqslant_p L_i$ mais $L_i \nleqslant_p L_{i+1}$. i.e. L_{i+1} est plus « simple » que L_i. □

4 Quelques remarques

On a donné une preuve du théorème de Ladner mais le langage L' que l'on a construit n'est pas du tout naturel. Jusqu'à maintenant, aucun problème naturel n'est connu être **NP**-intermédiaire bien que quelques-uns sont conjecturés dans cette classe **NPI** (par exemple, le problème de factorisation et le problème d'isomorphisme de graphes).

Il y a, en fait, une version plus forte de ce lemme 2 qui dit si $L \in \mathbf{NP} \setminus \mathbf{P}$, alors il existe un langage $L' \in \mathbf{NP} \setminus \mathbf{P}$ tel que L' se réduit en temps polynomial à L et L n'est pas Cook-réductible [1] à L'. La construction de L' consiste à enlever une partie de L pour empêcher toute machine en temps polynomial de décider L' et en même temps empêcher toute machine en temps polynomial avec l'oracle L' de décider L.

Références

[1] S. Arora and B. Barak : *Computational Complexity, A Modern Approach*, Cambridge University Press, 2009

[2] O. Goldreich : *Computational Complexity, A Conceptual Perspective*, Cambridge University Press, 2008

[3] L. Fortnow : *Two Proofs of Ladner's Theorem*, disponible sur `http://weblog.fortnow.com/media/ladner.pdf`

1. La Cook-réduction est une généralisation de la réduction en temps polynomial : A est Cook-réductible à B si et seulement si A est décidable par une machine en temps polynomial avec l'oracle B