Cours 5: Problèmes complets

Olivier Bournez

bournez@lix.polytechnique.fr
LIX, Ecole Polytechnique

Au menu

Problèmes complets

- La notion de réduction, basée sur le temps polynomial, ne s'avère pas pertinente pour parler de problèmes complets pour des classes incluses dans P.

- Nous allons introduire une notion de réduction plus fine basée sur l'espace logarithmique :

Sous menu

Fonction calculable en espace logarithmique

- Soient M et N deux alphabets.

- Une fonction $f : M^* \to N^*$ est *calculable en espace logarithmique* s'il existe un algorithme A, tel que pour tout mot w, A avec l'entrée w termine en utilisant un espace $\mathcal{O}(\log(n))$, où $n = |w|$, avec le résultat $f(w)$.

- On utilise toujours la convention que l'on ne compte pas l'entrée, w qui est en lecture seulement, et la sortie $f(w)$ qui est en écriture seulement dans l'espace utilisé.

- Proposition

  Une fonction calculable en espace logarithmique est calculable en temps polynomial.

- Proposition (Stabilité par composition)

  La composée de deux fonctions calculables en espace logarithmique est calculable en espace logarithmique.

Preuve

- Pour calculer la composition de f et de g calculables en espace logarithmique, on ne peut pas, sur une entrée w, calculer $w' = f(w)$, puis $g(w')$ car $w' = f(w)$ peut être de taille trop importante.

- Cependant, chaque pas de calcul de $g(w')$ nécessite au plus de lire une lettre de w', disons la lettre numéro i de w'.

- Or, étant donné le numéro i de cette lettre en binaire, on peut déterminer la valeur de cette lettre en espace logarithmique : simuler le calcul de f tout en maintenant un compteur qui compte le nombre de symboles j écrits jusque là, jusqu'à ce que $j = i$.

- En maintenant la valeur de i, on simule le calcul de g sur $w' = f(w)$ sans jamais écrire complètement w' :
  - à chaque fois que l'on a besoin de lire une nouvelle lettre de w', on utilise la procédure ci-dessus pour déterminer la valeur de cette lettre.

Réduction en espace logarithmique

- Soient A et B deux problèmes d'alphabet respectifs M_A et M_B, et de langages respectifs M_A et M_B.

- Une *réduction logspace de A vers B*, ou encore *réduction en espace logarithmique*, est une fonction $f : M_A^* \rightarrow M_B^*$ calculable en espace logarithmique telle que

$$w \in L_A \text{ ssi } f(w) \in L_B.$$

- On note $A \leq_L B$ lorsque A se réduit à B de cette façon.

Corollaire

Soient A et B deux problèmes. Supposons $A \leq_L B$. Alors $A \leq B$.

Theorem

$\leq_L$ est un préordre :

1. $L \leq_L L$
2. $L_1 \leq_L L_2$, $L_2 \leq_L L_3$ *implique* $L_1 \leq_L L_3$.

- Proposition (Réduction)

  Si $A \leq_L B$, et si $B \in \mathrm{P}$ alors $A \in \mathrm{P}$.

- Proposition (Réduction)

  Si $A \leq_L B$, et si $A \notin \mathrm{P}$ alors $B \notin \mathrm{P}$.

  - Les mêmes résultats sont vrais si on remplace P par les classes de complexité $\mathrm{NLOGSPACE}$, NP, PSPACE dans les propositions précédentes.

$\mathcal{C}$-complétude

- Soit $\mathcal{C}$ une classe de langages.
- Un problème A est dit $\mathcal{C}$-complet pour $\leq_L$ si
  1. il est dans la classe $\mathcal{C}$
  2. tout autre problème B de la classe $\mathcal{C}$ est tel que $B \leq_L A$.

Sous menu

Problèmes complets

- Réduction en espace logarithmique
- Retour sur la NP-complétude
- Un problème PSPACE-complet
- Un problème P complet
- Un problème $\mathrm{NLOGSPACE}$-complet

- Theorem
  Chacun des problèmes NP*-complets évoqués dans le chapitre précédent sont en fait* NP*-complets pour* $\leq_L$ *et pas seulement pour* $\leq$.

  - Lorsqu'on parlera de complétude dans la suite, on parlera toujours de complétude pour $\leq_L$.

Preuve

- On peut vérifier que dans chacune des preuves, la fonction de réduction utilisée est en fait calculable non seulement en temps polynomial mais en espace logarithmique.

Sous menu

Un problème pas très naturel

Theorem

Le problème

$$K = \{ < A, w, \mathbf{1}^n > \mid \textit{l'algorithme}$$

$$\textit{A accepte l'entrée w en espace n} \}$$

est PSPACE*-complet.*

Preuve

- Le problème est dans PSPACE car sur une entrée $< A, w, \mathbf{1}^n >$ il suffit de simuler l'algorithme A et de vérifier qu'il accepte w en espace n, ce qui se fait en espace linéaire.

- Soit L un problème de PSPACE.
  - L est accepté par un algorithme en espace $p(n)$ pour un certain polynôme p.
  - L se réduit à K par la fonction qui à un mot w associe le triplet $< A, w, \mathbf{1}^{p(|w|)} >$.
  - Cette fonction est bien calculable en espace logarithmique en la longueur de w.

QCIRCUITSAT

- Problème QCIRCUITSAT :
  - étant donné un circuit booléen $C(x_1, \cdots, x_n)$ à n variables,
  - peut-on déterminer s'il existe une valeur pour x_1 telle que pour toute valeur de x_2, il existe une valeur pour la variable x_3 telle que pour toute valeur de x_4 il existe une valeur pour la variable x_5 telle que toute valeur de x_6 ...etc, telles que $C(x_1, x_2, \cdots, x_n) = 1$?

- On peut toujours insérer des variables "stupides" qui n'interviennent pas dans le circuit C pour se ramener à une alternance stricte :

  Par exemple $\forall x_1 \forall x_2 \exists x_3 \ C(x_1, x_2, x_3) = 1$ peut se réécrire

  $\exists x \forall x_1 \exists y \forall x_2 \exists x_3 \ C(x_1, x_2, x_3) = 1.$

Theorem

Le problème $\mathrm{QCIRCUITSAT}$ *est* PSPACE*-complet.*

Preuve

- Le problème $\mathrm{QCIRCUITSAT}$ est dans PSPACE car il est facile de construire un algorithme récursif qui le résout :

  - pour une formule de la forme $\exists x_1 \phi$ (respectivement : $\forall x_1 \phi$) on fixe la valeur de x_1 à 0, on propage cette valeur pour x_1 dans ϕ pour obtenir $\phi_{x_1=0}$,
  - et on s'appelle récursivement pour savoir si la formule $\phi_{x_1=0}$ est satisfiable.
  - On fixe alors la valeur de x_1 à 1, on propage cette valeur pour x_1 dans ϕ pour obtenir $\phi_{x_1=1}$,
  - et on s'appelle récursivement pour savoir si la formule $\phi_{x_1=1}$ est satisfiable.
  - On accepte si et seulement si $\phi_{x_1=0}$ ou (resp. et) $\phi_{x_1=1}$ sont satisfiables.

  - L'algorithme fonctionne avec $\mathcal{O}(n)$ appels récursifs, chacun utilisant un espace constant, et donc en espace total $\mathcal{O}(n)$, où n est la taille de la formule.

- Pour montrer qu'il est complet, on va utiliser le principe de la preuve du théorème de Savitch.

  - Supposons que le problème L soit accepté en espace polynomial.
  - On va écrire par un circuit quantifié $C_i([X],[X'])$ le fait qu'il existe un chemin de longueur inférieure à 2^i dans le graphe G_w entre les configurations X et X'. Le graphe G_w est de taille $2^{p(n)}$ pour un certain polynôme p, où $n = |w|$.

  - Ce circuit s'obtient par récurrence sur i. Pour $i = 0$, c'est le circuit qui donne les arcs du graphe G_w.
  - Pour $i > 0$, on a envie d'écrire $C_i([X],[X'])$ comme

$$\exists [X''] \ C_{i-1}([X],[X'']) \wedge C_{i-1}([X''],[X']).$$

    Cependant si l'on fait ainsi, le circuit obtenu sera de taille exponentielle en i.

- L'idée est de quantifiée en réutilisant l'espace comme dans la preuve du théorème de Savitch : on écrit $C_i([X],[X'])$ comme

$$\exists[X'']\forall[D_1]\forall[D_2]\ C_i'([X],[X'],[D_1],[D_2],[X''])$$

où $C_i'([X],[X'],[D_1],[D_2],[X''])$ teste la relation

$$(([D_1]=[X]\wedge[D_2]=[X''])\vee(([D_1]=[X'']\wedge[D_2]=[X'])$$

$$\Rightarrow C_{i-1}([D_1],[D_2]).$$

- Cette fois la taille de C_i sera de l'ordre de celle de C_{i-1} plus $\mathcal{O}(p(n))$. Un mot w est dans le langage L si et seulement si $C_{p(n)}([X[w]],[X^*])$.

- La fonction qui à w associe le circuit quantifié $C_{p(n)}([X[w]],[X^*])$ est bien calculable en espace logarithmique.

- Le problème QSAT (aussi appelé QBF) consiste étant donnée une formule du calcul propositionnel en forme normale conjonctive ϕ avec les variables $x_1, x_2, \cdots, x_n$ (c'est-à-dire un instance de SAT) à déterminer si $\exists x_1 \forall x_2 \exists x_3 \cdots \ \phi(x_1, \cdots, x_n)$?

Theorem

Le problème QSAT *est* PSPACE*-complet.*

Preuve

- On utilise le fait que la satisfiabilité d'un circuit peut se ramener à celui d'une formule du calcul propositionnel en forme normale conjonctive :
  - on introduit une variable booléenne par porte du circuit.

Jeux stratégiques

- Le jeu GEOGRAPHY consiste à se donner un graphe orienté $G = (V, E)$.
  - Le joueur 1 choisit un sommet u_1 du graphe.
  - Le joueur 2 doit alors choisir un sommet v_1 tel qu'il y ait un arc de u_1 vers v_1.
  - C'est alors au joueur 1 de choisir un autre sommet u_2 tel qu'il y ait un arc de v_1 vers u_2, et ainsi de suite.
  - **On a pas le droit de repasser deux fois par le même sommet.**
  - Le premier joueur qui ne peut pas continuer le chemin $u_1 v_1 u_2 v_2 \cdots$ perd.

- Le problème GEOGRAPHY consiste à déterminer étant donné un graphe G et un sommet de départ pour le joueur 1, s'il existe une stratégie gagnante pour le joueur 1.

- Theorem

  Le problème GEOGRAPHY *est* PSPACE*-complet.*

Sous menu

Problèmes complets

- Réduction en espace logarithmique
- Retour sur la NP-complétude
- Un problème PSPACE-complet
- Un problème P complet
- Un problème $\mathrm{NLOGSPACE}$-complet

- Rappelons que le problème $\mathrm{CIRCUITVALUE}$ consiste, étant donnés un circuit $C(x_1, \cdots, x_n)$, et un ensemble de valeurs booléennes pour ses variables $\overline{x} \in \{0,1\}^n$, à déterminer si $C(\overline{x}) = 1$.

- **Theorem**
  Le problème $\mathrm{CIRCUITVALUE}$ *est* P*-complet.*

Preuve

- On a déjà vu dans le chapitre précédent que le problème $\mathrm{CIRCUITVALUE}$ était dans P.

- Soit L un langage de P.

  - On sait qu'il existe une famille de circuits de taille polynomiale $(C_n)_{n\in\mathbb{N}}$ qui reconnaît L.
  - L se réduit au problème $\mathrm{CIRCUITVALUE}$ par la fonction qui à un mot w associe $<< C_{|w|} >, w >$
  - cette fonction s'avère bien calculable en espace logarithmique.

- Le problème $\mathrm{SATVALUE}$ consiste, étant donnés une formule du calcul propositionnel en forme normale conjonctive ϕ avec les variables $x_1, x_2, \cdots, x_n$ (c'est-à-dire un instance de SAT), et un vecteur de valeurs $\overline{x} \in \{0,1\}^n$, à déterminer la valeur de $\phi(\overline{x})$.

- Theorem

  Le problème $\mathrm{SATVALUE}$ *est* P*-complet.*

Sous menu

Problèmes complets

- Réduction en espace logarithmique
- Retour sur la NP-complétude
- Un problème PSPACE-complet
- Un problème P complet
- Un problème $\mathrm{NLOGSPACE}$-complet

Theorem

Le problème REACH *est* $\mathrm{NLOGSPACE}$*-complet.*

Preuve

- Nous avons déjà vu que le problème REACH était dans $\mathrm{NLOGSPACE}$.
- Montrons qu'il est complet. Soit L un langage de $\mathrm{NLOGSPACE}$. L se réduit au problème REACH par la fonction qui à un mot w associe l'instance $< G_w, X[w], X^* >$: cette fonction est bien calculable en espace logarithmique.