Corrigé TD 5
Complexité

Luc Lapointe
luc.lapointe@ens-paris-saclay.fr
home.lmf.cnrs.fr/LucLapointe/

Exercice 1 – Planification

On note $I = [\![1, n]\!]$. Un *problème de planification* est donné par :
- n variables booléennes $\{x_i\}_{i \in I}$,
- m *opérations*, où une opération est définie par :
  - une condition de la forme $\bigwedge_{i \in I'} (x_i = \alpha_i)$ avec $I' \subseteq I$,
  - et une mise à jour de la forme $\{x_i \leftarrow \beta_i\}_{i \in I''}$ avec $I'' \subseteq I$.

  Voici un exemple d'opération : Si $x_1 = \top \wedge x_3 = \bot$ alors $x_1 \leftarrow \bot; x_2 \leftarrow \top$.
- Une configuration initiale s_{init} et une configuration finale s_{fin}.
- Une opération est applicable à une configuration si la condition de l'opération s'évalue à $\top$. Son application consiste à effectuer les mises à jour. Par exemple, l'opération précédente est applicable à la configuration $(\top, \bot, \bot)$ et conduit à la configuration $(\bot, \top, \bot)$.

Le problème consiste à déterminer s'il existe une suite d'applications des opérations, avec éventuellement plusieurs applications d'une même opération, qui conduit de la configuration initiale à la configuration finale.

1. Montrer que le problème de planification est dans PSpace.
2. Montrer que le problème de planification est PSpace-dur. Pour cela, on peut exploiter le problème ci-dessous qui est PSpace-dur :

Entrée: Le code d'une machine de Turing M, un mot w, un entier t écrit en unaire.
Question: Est-ce que M accepte w en espace t ?

Pour simplifier, on supposera que M a une seul bande de travail, et l'efface à l'issue du calcul (une fois dans l'état acceptant ou rejetant) ainsi que replace les têtes de lecture au début des bandes correspondantes.

Idée de solution

1. Utiliser PSpace = NPSpace. Deviner une suite de au plus 2^n opérations, car une suite d'opérations de taille minimale ne passe pas deux fois par la même configuration; et maintenir un tableau de taille n pour se souvenir des modifications.
2. Soit $M = (\Sigma, Q, q_i, \delta)$ une machine en entrée du problème suggéré. Utiliser $t * |\Sigma|$ variables qui représentent si la lettre $l \in \Sigma$ est à la position $i \in I$.

Cette valeur est bien calculable en espace logarithmique en la taille de M et t. Ensuite, rajouter des mises à jour pour chaque coordonnées possibles, pour chaque transition dans δ. C'est également faisable en espace logarithmique, en comptant jusqu'à t et jusqu'à $|\delta|$.

Exercice 2 – Géographie

Le jeu Géographie se joue de la manière suivante :

- Le jeu commence avec un nom de ville donné, par exemple *Gometz-le-Châtel*.
- La première joueuse donne le nom d'une ville dont la première lettre coïncide avec la dernière lettre de la ville précédente, par exemple *Les Ulis*.
- Le deuxième joueur donne un autre nom de ville, commençant également par la dernière lettre de la ville précédente, par exemple *Saclay*.
- Le premier joueur joue à nouveau et ainsi avec la restriction qu'il est interdit de donner le nom d'une ville déjà vue dans le jeu.
- La première personne qui ne peut pas trouver de nom de ville a perdu.

Ce jeu peut être décrit à l'aide d'un graphe orienté dont les noeuds représentent les villes et où une arête (X, Y) signifie que la dernière lettre de X est la première lettre de Y. Ce graphe a également un noeud distingué qui correspond à la ville initiale. Une partie consiste à se déplacer d'un sommet courant en choisissant alternativement l'arête à emprunter, en s'interdisant de repasser par un sommet déjà visité, et il faut réussir à bloquer l'autre.

Géographie Généralisée, abrégé GG, correspond au problème suivant :

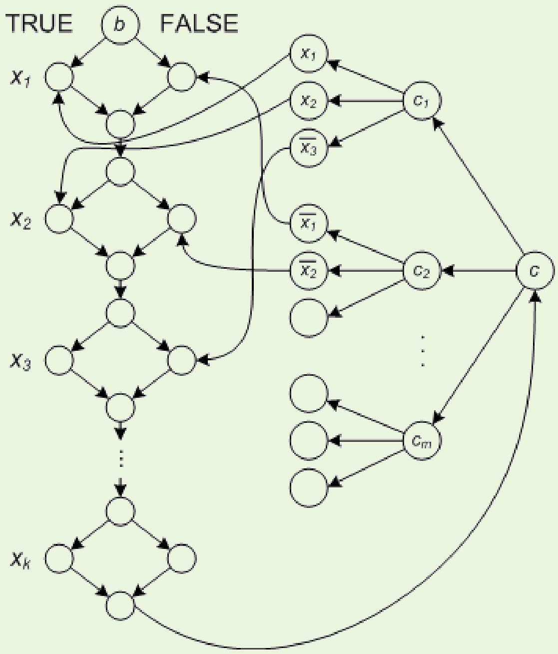
Entrée: un graphe orienté G, un noeud initial s.

Question: La joueuse 1 a une stratégie gagnante pour GG sur G depuis s ?

1. Montrer que GG est dans PSpace.
2. Montrer que GG est PSpace-dur via une réduction depuis QBF.

Idée de solution

1. Faire du backtracking, en gardant en mémoire les sommets déjà visités, et en énumérant à chaque moment où il faut faire un choix avec par exemple l'ordre lexicographique. Les feuilles de l'arbre du backtracking sont étiquetées gagnantes ou pas selon à qui c'est de jouer, puis les noeuds sont étiquetés en partant du bas en fonction de si c'est le tour de la joueuse 1 et qu'il existe un coup qui amène à un sommet gagnant, ou du joueur 2 et que tous les coups amènent à un sommet gagnant. On ne retient jamais tout l'arbre en mémoire, seulement une branche.
2. Utiliser une construction inspirée de celle ci-dessous (prise sur Wikipedia)



La construction dessinée fonctionne pour une entrée de QBF où les quantificateurs universels et existentiels s'alternent. Si à un moment deux quantificateurs universels ou existentiels se suivent, il est possible de rajouter un sommet d'arité 1 pour que ça soit le bon joueur qui joue.

L'idée de cette construction est de passer dans le chemin dans les variables par les valeurs qui ne seront **pas** prises. Une fois toutes les valeurs sélectionnées, l'adversaire choisit une clause qui doit être validée, puis la joueuse 1 montre la variable qui est vraie.

Exercice 3 – Espace polylogarithmique

On rappelle le théorème de hiérarchie en espace :

Théorème *Pour deux fonctions constructibles en espace f et g telles que $f(n) = o(g(n))$, on a $\mathrm{DS{\small PACE}}\,(f) \subsetneq \mathrm{DS{\small PACE}}\,(g)$.*

On définit à présent la classe de complexité :

$$\mathrm{P{\small OLY}L{\small OG}} = \bigcup_{k>0} \mathrm{SPACE}\left(\log^k(n)\right)$$

1. Montrer que PolyLog n'a pas de problème complet pour des réductions en espace logarithmique. Que peut-on en déduire quant à la comparaison entre les classes PTime et PolyLog ?
2. On rappelle que $\mathrm{PS{\small PACE}} = \bigcup_{k>0} \mathrm{SPACE}\left(n^k\right)$. Est-ce que PSpace a un problème complet pour des réductions en espace logarithmique ? Pourquoi est-ce que la preuve de la question précédente ne s'applique pas à PSpace ?

Idée de solution

1. cf document joint sur ma page.
   En particulier, on ne sait pas comment PolyLog se situe par rapport à PTime !
2. QBF est PSpace-complet. La preuve jointe exploite le fait que la sortie de la réduction est plus grande mais son degré dans le polynôme ne change pas. C'est un argument valable dans PolyLog mais pas dans PSpace.