Complexité - TD 4

Benjamin Bordais
bordais@lsv.fr
`www.lsv.fr/~bordais/`

9 Décembre 2021

On rappel la définition de la classe de complexité PSPACE : un langage (ou problème de décision) est dans PSPACE s'il existe une machine de Turing M déterministe décidant le langage L qui termine sur toute entrée et telle qu'il existe un polynôme p tel que l'espace pris par la machine M sur toute en entrée x de taille n prend un espace au plus $p(n)$ (en particulier, le temps pris peut être exponentiel en n).

Question 1 (Mon tout premier problème PSPACE-complet) *Montrer que le problème suivant est* PSPACE-*complet :*

— *Entrée : le code d'une machine de Turing M, un mot w, une quantité d'espace t écris en unaire*
— *Sortie : M accepte w en espace t*

Solution 1 *Prouvons d'abord que ce problème est dans* PSPACE. *Considérons une instance (M, w, t) du problème. Il faut remarquer que l'on ne peut pas se contenter de simuler M sur w tout en s'assurant que l'espace pris ne dépasse pas t car alors on n'est pas garanti que l'algorithme termine. Pour cela, il faut rajouter un timeout qui dépendra de M, t et x. En effet, pour une machine M fonctionnant sur un alphabet Σ, sur un ensemble d'états Q, sur une entrée x de taille n, avec un espace pris inférieur à t, le nombre total de configurations possibles est inférieur à $f(n) = |Q| \cdot |\Sigma|^t \cdot t \cdot n$ (les deux derniers termes faisant référence aux têtes de lecture). L'algorithme consiste alors alors à simuler M sur x en s'assurant que l'espace pris n'excède pas t et le nombre d'étapes pris ne dépasse pas $f(n)$. A noter que $f(n)$ peut être représenté en espace polynomial, tout comme un état courant du calcul de M sur x.*

Pour ce qui est de la PSPACE-*dureté, si l'on considère un langage L dans* PSPACE *décidé en espace polynomial p par une machine de Turing M, on considère alors la réduction f telle que $f(x) = (M, x, p(|x|))$. On a alors $x \in L$ si et seulement si $f(x)$ est dans notre langage. De plus, f est constructible en espace logarithmique étant donné qu'il suffit juste d'écrire un polynome sur la bande de sortie (le polynome est fixé).*

Un problème PSPACE-complet classique est le suivant, appelée QBF (pour *quantified boolean formula*) :

Entrée : une formule bouléenne quantifié $\Phi = Q_1 x_1, \ldots, Q_n x_n : \varphi(x_1, \ldots, x_n)$ avec φ une formule de la logique propositionnelle sans variable libre et Q_i un quantificateur $\forall$ ou $\exists$.

Sortie : oui si et seulement si la formule est satisfiable (avec interprétation des quantificateurs avec la sémantique habituelle)

Question 2 *Qu'est ce que l'on obtient si tous les quantificateurs sont existentiels ? Et s'ils sont tous universels ?*

Solution 2 *S'ils sont tous existentiels, on obtient* SAT *et le problème est alors* NP*-complet. S'ils sont tous universels, on obtient le dual, et le problème est co*NP*-complet.*

Jeu et complexité

Question 3 *Un jeu (à deux joueurs) à tour est un graphe orienté $G = (V, E)$ où l'ensemble des sommets $V = V_A \uplus V_B$ est partitionné en sommets appartenant au joueur A (i.e. V_A) et les sommets appartenant au joueur B (i.e. V_B) avec un sommet distingué $v_0 \in V$ qui correspond au sommet initial. Le graphe est tel que chaque sommet a un successeur, c'est-à-dire $\mathsf{Succ}(v) = \{v' \in V \mid (v, v') \in E\} \neq \emptyset$ pour tout $v \in V$. Une partie correspond alors à un chemin fini ou infini $\rho = v_0 \cdot v_1 \cdots \in V^* \cup V^\omega$ avec v_0 le sommet initial. Si la partie en est au sommet $v_i \in V_A$ alors c'est au joueur A de choisir le prochain sommet $v_{i+1} \in \mathsf{Succ}(v_i)$, tandis que c'est au joueur B si $v_i \in V_B$. Une condition de victoire détermine alors lorsqu'une partie infinis est gagnés par le joueur A (on considère des jeux perd/gagne, ainsi si le joueur A ne gagne pas, alors c'est le joueur B qui gagne). Un joueur $C \in \{A, B\}$ a une stratégie gagnante (ou gagne) à partir d'un sommet $v \in V$ s'il peut choisir le prochain sommet à partir de n'importe quelle partie se terminant en V_C tel que chaque partie infinie cohérente avec ces choix est gagnée par le joueur C.*

1. *Supposons que la condition de victoire soit une condition d'atteignabilité : étant donné un ensemble cible $T \subseteq V$, le joueur A gagne si et seulement si un sommet de T est vu. Montrer que décider le vainqueur d'un jeu d'atteignabilité à partir du sommet $v_0 \in V$ peut se faire en temps polynomial.*

   Indice : construire inductivement l'ensemble des sommets à partir desquels le joueur A peut forcer à se rapprocher de la cible T (c'est ce que l'on appelle l'attracteur de T).
2. *Considérons un $k \in \mathbb{N}$. Une condition d'atteignabilité k-généralisé est définie comme suit : étant donné k ensembles cibles $T_1, \ldots, T_k \subseteq V$, le joueur A gagne si et seulement si, pour tout $1 \leq i \leq k$, un sommet de T_i est vu. Montrer que décider le vainqueur d'un jeu d'attignabilité k-generalisé à partir du sommet $v_0 \in V$ peut être fait en temps polynomial.*
3. *Une condition d'atteignabilité généralisé est définit comme suit : étant donné plusieurs ensembles cibles $T_1, \ldots, T_k \subseteq V$, le joueur A gagne si et seulement si, pour tout $1 \leq i \leq k$, un sommet de T_i est vu. La différence avec l'objectif d'atteignabilité k-generalisé est que le nombre de cibles n'est pas borné a priori. Montrer que décider le vainqueur d'un jeu d'atteignabilité généralisé est* PSPACE*-complet.*

   Indice : pour l'appartenance à PSPACE, on pourra utiliser que si le joueur A peut gagner avec k cibles, il peut voir toutes les cibles en au plus $k \cdot n$ étapes avec $n := |V|$.
4. *Montrer que décider le vainqueur d'un jeu d'atteignabilité généralisé lorsque $V_B = \emptyset$ (i.e. le joueur A joue tout seul) est* NP*-complet.*

Solution 3

1. *Considérons le jeu d'atteignabilité $G = (V, E)$ et un ensemble cible $T \subseteq V$. Définissons inductivement la séquence d'ensembles de sommets $(X_i)_{i \in \mathbb{N}} \subseteq V^{\mathbb{N}}$ avec $X_0 := T$ et, pour tout $i \geq 0$, on a :*

$$X_{i+1} := X_i \cup \{v \in V_A \mid \mathsf{Succ}(v) \cap X_i \neq \emptyset\} \cup \{v \in V_B \mid \mathsf{Succ}(v) \subseteq X_i\}$$

   On pose alors $X := \cup_{i \in \mathbb{N}} X_i \subseteq V$. Alors, le joueur $\mathcal{A}$ gagne si et seulement si $v_0 \in X$.

D'abord, pour tout $x \in X$, on dénote par $i(x)$ le plus petit indice $i \in \mathbb{N}$ tel que $x \in X_i$. C'est-à-dire $i(x) := \min\{i \in \mathbb{N} \mid x \in X_i\}$. Supposons que le joueur A choisisse un successeur tel que, pour tout $v \in X \cap V_A$ avec $i(v) > 0$, le successeur choisi $v' \in V$ est tel que $v' \in X$ et $i(v') < i(v)$ (ce qui possible par définition de X). Remarquons que, par définition de X et i, pour tout $v \in X \cap V_B$ tel que $i(v) > 0$, on a $\mathsf{Succ}(v) \subseteq \{v' \in X \mid i(v') < i(v)\}$. Dans ce cas, toute partie $\rho = x_0 \cdot x_1 \cdots$ commençant dans un sommet $v \in X$ tel que $i(x_0) > i(x_1) > \cdots$ jusqu'à un sommet $x_i \in X$ est tel que $i(x_i) = 0$. C'est-à-dire, $x_i \in X_0 = T$. Il vient que le joueur A gagne à partir de tous le sommets de X.

De la même manière, montrons que le joueur B gagne à partir de n'importe sommet qui n'est pas dans X. Supposons que le joueur B joue tel que, pour tout $v \in (V \setminus X) \cap V_B$, le successeur choisi $v' \in V$ est tel que $v' \in V \setminus X$ ce qui est possible par définition de X. Remarquons que, pour tout $v \in (V \setminus X) \cap V_A$, on a $\mathsf{Succ}(v) \subseteq (V \setminus X)$ par définition de X. A présent, considérons une partie $\rho = x_0 \cdot x_1 \cdots$ commençant d'un sommet $v_0 \in V \setminus X$. On a que, pour tout $i \in \mathbb{N}$, si $x_i \notin X$ alors $x_{i+1} \notin X$ et $v_0 \notin X$. C'est-à-dire, aucun sommet de ρ n'est dans X, en particulier, on n'atteint jamais T. Ainsi, le joueur B gagne à partir de n'importe quel sommet qui n'est pas X.

Au final, on a que le joueur A gagne depuis v_0 si et seulement si $v_0 \in X$. De plus, l'ensemble X peut être calculé en temps polynomial (puisque cela nécessite $|V|$ boucles sur les sommets qui ne sont pas dans X). Ainsi, déterminer le vainqueur d'un jeu d'atteignabilité peut se faire en temps polynomial.

2. *On réduit ce problème au problème précédent. C'est-à-dire, on ajoute une 'mémoire' à chaque sommet pour stocker l'ensemble des ensembles de sommets qui ont déjà été vus. Spécifiquement, considérons un graphe $G = (V, E)$ et k cibles $T_1, \ldots, T_k \subseteq V$. Pour tous sommets $v \in V$, on dénote par $t(x) \subseteq \{1, \ldots, k\}$ l'ensemble des ensembles cibles auxquels il appartient : $t(x) := \{i \leq k \mid x \in T_i\}$. A présent, considérons le jeu d'atteignabilité suivant $G_k = (V_k, E_k)$ avec $T \subseteq V$ tel que :*
   — $V_k := V \times 2^{\{1,\ldots,k\}}$;
   — $E_k := \{((v, s), (v', s')) \mid (v, v') \in E \wedge s' = s \cup t(v)\}$.
   — $T := \{(v, \{1, \ldots, k\}) \mid v \in V\}$
   Avec cette construction, n'importe quelle partie à partir d'un sommet $v \in V$ peut être traduit en une partie de G_k à partir de $(v, \emptyset) \in V_k$, et réciproquement. Il vient que le joueur A gagne dans G à partir de v_0 le jeu d'atteignabilité k-généralisé si et seulement si il gagne dans le jeu jeu d'atteignabilité G_k depuis $(v_0, \emptyset)$. Cela peut être décidé $O(|G_k|) = O(2^k \cdot |G|) = O(|G|)$ car k est une constante.

3. *Dénotons par* GenReach *ce problème de décision. Montrons tout d'abord que c'est dans* PSPACE. *Pour tous sommets $v \in V$, on dénote par $t(x) \subseteq \{1, \ldots, k\}$ l'ensemble des indices des ensembles cibles auxquels il appartient : $t(x) := \{i \leq k \mid x \in T_i\}$. On définit maintenant la fonction récursive* $\mathsf{win} : (v, \mathsf{visit}, n) \mapsto$ "true *si et seulement si le joueur A gagne dans le graphe $G = (V, E)$, à partir de $v \in V$, dans une version du jeu où les cibles de* visit *ont déjà été vu et les cibles restantes doivent être vus en au plus n étapes". A présent* win *est une simple fonction récursive :*

$$\mathsf{win}(v, \mathsf{visit}, n) = \begin{cases} \mathsf{true} & \textit{if } \mathsf{visit} = \{1, \ldots, k\} \\ \mathsf{false} & \textit{otherwise, if } n = 0 \\ \exists (v, v') \in E,\ \mathsf{win}(v', \mathsf{visit} \cup h(v), n-1) = \mathsf{true} & \textit{otherwise, if } v \in V_A \\ \forall (v, v') \in E,\ \mathsf{win}(v', \mathsf{visit} \cup h(v), n-1) = \mathsf{true} & \textit{otherwise, if } v \in V_B \end{cases}$$

Dans ce cas, le joueur A gagne à partir de v_0 si et seulement si $\mathsf{win}(v_0, \emptyset, n \cdot k) =$ true.

Un algorithme implémentant cette procédure s'exécute en espace polynomial : on explore chaque successeur et on s'appelle récursivement. Le nombre d'appels imbriqué est au plus $n \cdot k$ et chaque appel prend un espace polynomial à stocker (le sommet courant, l'ensemble des ensemble de sommets déjà vus et l'indice n).

Considérons maintenant la réduction pour la PSPACE*-dureté. Soit $\Phi = Q_1 x_1, Q_2 x_2, \ldots, Q_n x_n, \phi$ une formule* QBF *avec $Q_i \in \{\forall, \exists\}$ pour tout $i \leq n$ et $\phi = \bigwedge_{1 \leq j \leq k} C_j$ une formule* CNF*. On construit le jeu d'atteignabilité généralisé suivant :*

— $G_\Phi := (V, E)$;
— $V := \{x_i, \neg x_i \mid 1 \leq i \leq n\} \cup \{s_j \mid 1 \leq j \leq n+1\}$;
— $V_B := \{s_i \mid Q_i = \forall\}$ *et* $V_A := V \setminus V_B$;
— $E := \{(s_i, x_i), (s_i, \neg x_i) \mid 1 \leq i \leq n\} \cup \{(x_i, s_{i+1}), (\neg x_i, s_{i+1}) \mid 1 \leq i \leq n\} \cup \{(s_{n+1}, s_{n+1})\}$;
— *Il y a k cibles $T_1, \ldots, T_k \subseteq V$ avec $T_i := \{l_j \in V \mid l_j \in C_i\}$ où l_j est un litéral : soit égal à x_j ou $\neg x_j$.*

D'abord remarquons que cela peut être implémenté en espace logarithmique car il s'agit simplement de boucler sur les variables et clauses de l'entrée pour générer la sortie.

Montrons que $\Phi \in$ QBF $\Leftrightarrow G_\Phi \in$ GenReach*. On le montre par induction sur les valuations partielles des variables. Pour tout $k \leq n$, une valuation k-partielle est une valuation $v_k : \{x_1, \ldots, x_k\} \to \{\top, \bot\}$ des variables $x_1, \ldots, x_k$. Pour chaque assignation partielle, on lui associe la partie correspondante ρ_{v_k} dans le graphe G_Φ à partir de s_1 vers s_{k+1} qui visite les variables x_i si $v_k(x_i) = \top$ et visite $\neg x_i$ sinon. On dénote par Φ_{v_k} la formule* QBF *que l'on obtient :*

$$\Phi_{v_k} = Q_{k+1} x_{k+1}, \ldots, Q_n x_n, \phi[v_k]$$

où $\phi[v_k]$ est la formule ϕ où chaque variable de $\{x_1, \ldots, x_k\}$ a été remplacé par sa valeur donné par v_k. De la même manière, on dénote par ${G_\Phi^{\rho_{v_k}}}$ le jeu qui commence en s_{k+1} avec le chemin ρ_{v_k} déjà vu. Montrons alors la propriété $\mathcal{P}(k)$ suivante par induction induction sur $0 \leq k \leq n$ avec $\mathcal{P}(k)$: pour toutes valuations k-partielles v_k, on a $\Phi_{v_k} \in$ QBF $\Leftrightarrow G_\Phi^{\rho_{v_k}} \in$ GenReach.

D'abord, $\mathcal{P}(n)$ est valide par définition des ensemble T_i, ils correspondent exactement aux clauses de la formule. Ainsi, toutes les clauses sont satisfaites par une valuation v_n si et seulemne tsi tous les ensembles cibles sont visitéspar la partie ρ_{v_n}. Supposons à présent que la propriété $\mathcal{P}$ est valide pour un $0 < k+1 \leq n$. Considérons une valuation k-partielle v_k, la formule QBF *Φ_{v_k} et le jeu correspondant $G_\Phi^{\rho_{v_k}}$. Supposons que $Q_k = \exists$. On a alors l'équivalence suivante :*

$$\begin{aligned}
\Phi_{v_k} \in \mathsf{QBF} &\Leftrightarrow \Phi_{v_k[x_{k+1} \to \top]} \in \mathsf{QBF} \vee \Phi_{v_k[x_{k+1} \to \bot]} \in \mathsf{QBF} \\
&\Leftrightarrow G_\Phi^{\rho_{v_k} \cdot x_{k+1} \cdot s_{k+2}} \in \mathsf{GenReach} \vee G_\Phi^{\rho_{v_k} \cdot (\neg x_{k+1}) \cdot s_{k+2}} \in \mathsf{GenReach} \\
&\Leftrightarrow G_\Phi^{\rho_{v_k}} \in \mathsf{GenReach}
\end{aligned}$$

Le cas $Q_k = \forall$ est analogue. C'est-à-dire $\mathcal{P}(k)$ est valide. En fait, c'est valide pour tout $0 \leq k \leq n$. En particulier, $\mathcal{P}(0)$ est valide, ce qui correspond exactement à l'équivalence $\Phi \in$ QBF $\Leftrightarrow G_\Phi \in$ GenReach.

4. *Le problème de décision que l'on obtient est maintenant dans* NP *: on peut deviner un chemin de longueur au plus $k \cdot n$ est on peut vérifier en temps polynomial*

qu'effectivement, tous les ensemble de sommets apparaissent. De plus, la réduction précédente s'applique aussi ici : en effet, SAT *est le cas particulier de* QBF *sans quantificateur universel. Dans la réduction, cela correspond au fait que l'ensemble de sommets V_B est vide, c'est-à-dire la restriction du problème que l'on considère. Ainsi, on a réduction logspace depuis* SAT*, le problème est donc* NP*-dur.*