Complexité - TD 3.2

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

2 Décembre 2021

On rappelle la définition du problème de décision SAT :
— ENTRÉE : une formule propositionnelle ϕ sous forme normale conjonctive
— SORTIE : ϕ est satisfiable
Ce problème est NP-complet. Le problème $3-\mathsf{SAT}$ est la restriction du problème SAT aux formules contenant au plus trois littéraux par clause. Il est aussi NP-complet.

Question 1 (Échauffement) *Soit $L \subseteq \Sigma^*$ un langage sur l'alphabet Σ. À quelles conditions L est* PTIME*-dur pour des réductions en temps polynomial ?*

Solution 1 *Tout langage L tel qu'il existe $w_i \in L$ et $w_o \notin L$ est* PTIME*-dur pour des réductions en temps polynomial. En effet, considérons un langage $L' \in$ PTIME. On pose alors la réduction $f : \Sigma^* \to \Sigma^*$ qui, sur une entrée $x \in \Sigma^*$, décide en temps polynomial si $x \in L'$ (ce qui est possible car $L' \in$ PTIME) et qui renvoie w_i (resp. w_o) si $x \in L'$ (resp. si $x \notin L'$). On a bien $x \in L' \Leftrightarrow f(x) \in L$ et la transformation f calculable en temps polynomial. Ainsi, tout langage non-trivial (c'est-à-dire différent de $\emptyset$ et Σ^*) est* PTIME*-dur pour des réductions en temps polynomial.*

C'est pour cela que lorsque l'on considère la dureté pour une classe, on considère des réductions qui sont moins coûteuses (en temps ou en espace) qu'un algorithme pour décider des problème de décision de cette classe. Ainsi, les langages PTIME*-dur sont usuellement considérés pour des réductions en espace logarithmique.*

Question 2 (Un simple problème NP-complet) *Montrer que le langage suivant est* NP*-complet : $\{(M, x, 1^t) \mid M$ accepte sur x en temps au plus $t\}$ avec M le code d'une machine de Turing non-déterministe.*

Solution 2 *Il faut d'abord montrer que $L \in$ NP. On considère l'algorithme non déterministe qui, sur une entrée $(M, x, 1^t)$ simule M sur x (en simulant le non-déterminisme de M) en conservant un time-out qui compte le nombre d'étape effectué. Si ce time-out dépasse t, cet algorithme rejette. Sinon, on imite ce qu'aurait fait la machine M. Cet algorithme décide bien L et est en temps polynomial car le time-out t est donné en unaire dans l'entrée (ainsi, avoir un compteur allant jusqu'à t correspond bien à un temps polynomial en la taille de l'entrée). Ainsi, $L \in$ NP.*

Montrons à présent de L est NP*-dur. Soit $L' \in$ NP. Il existe alors une machine de Turing non déterministe M et un polynôme p tels que M décide L et termine en temps au plus $p(n)$ où n est la taille de l'entrée. On construit alors une réduction $f : \Sigma^* \to \Sigma^*$ ainsi : pour $x \in \Sigma^*$, on a $f(x) = (M, x, 1^{p(|x|)})$. Par définition de L et M, on a bien $x \in L' \Leftrightarrow f(x) \in L$. De plus f peut se calculer en espace logarithmique : la machine M est fixé on la recopie sur la bande de sortie en temps constant, on recopie ensuite l'entrée x et*

on a ensuite besoin d'un compteur (en binaire) de taille $\log(p(|x)) = O(\log(|x|))$. *Ainsi,* L *est* NP*-dur pour des réduction en espace logarithmique.*

La classe coNP

Soit une classe $\mathcal{C}$ de problèmes de décision. La classe co$\mathcal{C}$ correspond à l'ensemble des langages L tel que $\bar{L} \in \mathcal{C}$: co$\mathcal{C} = \{\bar{L} \mid L \in \mathcal{C}\}$.

Question 3 *Supposons que le langage* L *soit complet pour la classe* $\mathcal{C}$*. Exhiber un langage complet pour la classe* co$\mathcal{C}$*.*

Solution 3 *Le langage* $\bar{L}$ *est complet pout la classe* co$\mathcal{C}$*. En effet,* $\bar{L} \in$ co$\mathcal{C}$ *par définition. De plus, soit* $L' \in$ co$\mathcal{C}$*. On a* $\bar{L'} \in \mathcal{C}$*. Comme* L *est complet pour la classe* $\mathcal{C}$*,* $\bar{L'}$ *peut se réduire à* L*. C'est-à-dire, il existe une fonction* f *calculable en espace logarithmique telle que* $x \in \bar{L'} \Leftrightarrow f(x) \in L$*. On a alors* $x \in L' \Leftrightarrow \neg(x \in \bar{L'}) \Leftrightarrow \neg(f(x) \in L) \Leftrightarrow f(x) \in \bar{L}$*. Ainsi,* f *est également une réduction de* L' *vers* $\bar{L}$*. Ainsi,* $\bar{L}$ *est dur pour la classe de complexité* $\mathcal{C}$*.*

Question 4 *Prouver que le problème de décision suivant est* coNP*-complet :*
Tautology *:*
- *ENTRÉE : un formule propositionelle* ϕ *sous forme normale disjonctive*
- *SORTIE : toute valuation* ν *satisfait* ϕ

Solution 4 *Montrons tout d'abord que ce problème est dans* coNP*. Le problème* $\overline{\text{Tautology}}$ *peut s'écrire ainsi :*
- *ENTRÉE : un formule propositionnelle* ϕ *sous forme normale disjonctive*
- *SORTIE : il existe une valuation* ν *qui ne satisfait pas* ϕ

On a $\overline{\text{Tautology}} \in$ NP *car il suffit de deviner une valuation et de vérifier (en temps polynomial) que celle-ci ne satisfait pas la formule. Montrons à présent que* Tautology *est* coNP*-dur. On sait, ar la question précédente, que le problème* $\overline{\text{SAT}}$ *est* coNP*-complet avec* $\overline{\text{SAT}}$ *:*
- *ENTRÉE : un formule propositionnelle* ϕ *sous forme normale conjonctive*
- *SORTIE : toute valuation* ν *ne satisfait pas* ϕ

Pour une formule $\phi = \wedge_{i=1}^{n}(\vee_{j=1}^{a_i} l_i)$*, on pose* $f(\phi) = \neg\phi = \vee_{i=1}^{n}(\wedge_{j=1}^{a_i} \neg l_i)$*. On a* $f(\phi)$ *sous forme normale disjonctive, calculable en espace logarithmique et* ϕ *n'est pas satisfiable si et seulement si* $f(\phi)$ *est une tautologie. C'est-à-dire,* $\phi \in \overline{\text{SAT}} \leftrightarrow f(\phi) \in$ Tautology*.*

Question 5 *Le problème* SAT *reste-t-il* NP*-complet si la formula est en forme normale disjonctive (au lieu de conjonctive) ? Et pour* Tautology *? (si la formula est en forme normale conjonctive au lieu de disjonctive)*

Solution 5 *A priori non. En effet, dans ce cas, le problème est dans* L*. Il suffit de parcourir les conjonctions pour en trouver une à satisfaire. Satisfaire une conjonction revient alors à chercher s'il y une contradiction au sein de cette conjonction (une variable apparaissant à la fois positivement et négativement). Il suffit juste de comparer les variables qui apparaissent dans la conjonction avec les autres. Cela demande plusieurs pointeurs, mais peut seulement un nombre constant. C'est la même chose pour le problème de* Tautology *en CNF.*

Question 6 *Un problème* coNP*-complet est-il (a priori) dans* NP *?*

Solution 6 *Si un problème* coNP*-complet était dans* NP*, on aurait que* NP = coNP *(ce qui est un problème ouvert ce jour). En effet, soit L un problème* coNP*-complet dans* NP*. Pour tout problème $L' \in$ coNP, L' se réduit à L en espace logarithmique, et comme* NP *est stable par réduction logarithmique, on en déduit que $L' \in$ NP. Ainsi,* coNP $\subseteq$ NP*. Pour $A \in$ NP, on a alors $\bar{A} \in$ coNP $\subseteq$ NP, donc $\bar{\bar{A}} = A \in$ coNP. C'est-à-dire,* NP $\subseteq$ coNP*. Ainsi, on a* coNP = NP*.*

Variantes de SAT

La première réduction peut se faire à partir de SAT. La deuxième peut se faire à partir de $3 - \mathsf{SAT}$

Question 7 *Prouver que le problème de décision suivant est* NP*-complet :*

MONOTONE $-$ SAT *:*

- *ENTRÉE : un formule propositionnelle ϕ sous forme normale conjonctive tel que, dans chaque clause, soit tous les littéraux positivement, soit ils apparaissent tous négativement (la formule est alors dite monotone)*
- *SORTIE :il existe une valuation satisfaisant ϕ*

Solution 7 *Le problème est dans* NP *car c'est un cas particulier de* SAT*. Considérons maintenant une instance $\varphi = \wedge_{i=1}^{n}(\vee_{j=1}^{m} l_{i,j})$ de* SAT*. Pour toutes variables x apparaissant dans φ, on introduit une nouvelle variable $\bar{x}$. On construit alors la formule $\psi = \varphi' \wedge (\wedge_{k=1}^{p}(\neg x_k \Leftrightarrow \bar{x_k}))$ où φ' correspond à la formule φ où toutes les instances de $\neg x$ ont été remplacées par $\bar{x}$ (pour toutes variables x). Les variables de φ étant dénotées $(x_k)_{k=1}^{p}$. Il vient que φ' est monotone (toues les clauses étant positives) et que ϕ et φ sont équisatisfiables. Il suffit ensuite de remarquer que $(\neg x \Leftrightarrow \bar{x}) \equiv (x \vee \bar{x}) \wedge (\neg x \vee \neg \bar{x})$. Ainsi, ψ peut s'écrire comme une conjonction de clauses monotone. On remarque finalement que cette transformation peut s'effectuer en espace logarithmique (avec une boucle sur les variables de φ).*

Question 8 *Prouver que le problème de décision suivant est* NP*-complet :*

NAE $-$ SAT *(pour 'not-all-equal') :*

- *ENTRÉE : un formule propositionnelle ϕ sous forme normale conjonctive (avec au plus 3 littéraux par clause)*
- *SORTIE : il existe une valuation tel que, dans chaque clause de ϕ avec au moins deux littéraux, il y a au moins un littéral satisfait et un littéral non-satisfait. Les clauses à un littéral ont seulement besoin d'être satisfait (une telle valuation nae-satisfait φ).*

Solution 8 *Le problème est dans* NP *car c'est un cas particulier de* SAT*. Considérons maintenant une instance $\varphi = \wedge_{i=1}^{n} C_i$ de* SAT*. On considère alors n nouvelles variables $w = (w_i)_{1 \leq i \leq n}$ et une autre nouvelle variable x_{F}. On construit alors la formule $\varphi' := (\neg x_{\mathsf{F}}) \wedge \wedge_{1 \leq i \leq n}(C_i^1 \wedge C_i^2)$. pour tout $1 \leq i \leq n$, pour $C_i = \alpha_i^1 \vee \alpha_i^2 \vee \alpha_i^3$, on pose :*

- $C_i^1 := \alpha_i^1 \vee \alpha_i^2 \vee w_i$;
- $C_i^2 := \alpha_i^3 \vee x_{\mathsf{F}} \vee \neg w_i$.

On peut ensuite vérifier que $\varphi(y, x)$ est satisfait si et seulement si $\exists w$, $\exists x_{\mathsf{F}} : \varphi'$ est nae-satisfait : Si $\alpha_i^1 \vee \alpha_i^2$ est satisfait, alors w_i est assigné à faux et si α_i^3 est satisfait alors w_i est assigné à vrai, et réciproquement. La réduction est logspace car il suffit de parcourir la les clauses et de les dupliquer. De plus, la formula obtenue a au plus 3 littéraux par clause.

Question 9 ((Bonus)) *Exhiber une réduction de* SAT *vers* $3 - \mathsf{SAT}$*.*