Examen du cours Complexité (L3)

Les documents (notes, polycopiés, ..) et calculatrices (téléphone, tablette, ..)
ne sont pas autorisés.

Date : 15 janv. 2024 à 10h00 / Durée : 2 heures

Exercice 1 : Machines à témoin

On appelle *machine à témoin* une machine de Turing M munie de deux bandes d'entrée (en lecture seule, non modifiables) en plus des bandes de travail habituelles. La première entrée, nommée x, est l'input de la machine comme d'habitude. La deuxième entrée, nommée y, est appelée « témoin » ou « certificat ». On note $M(x, y)$ le calcul de M sur ces deux entrées. Les notions de temps de calcul et d'espace de travail sont inchangées, et M peut être non déterministe. Nous avons vu une caractérisation des langages de NP qui peut s'exprimer avec les machines à témoin :

Thm. 1 (vu en cours). Un langage $L \subseteq A^*$ est dans NP si, et seulement si, il existe un polynôme p et une machine à témoin *déterministe* en temps polynomial M telle que pour tout $x \in A^*$

$$x \in L \Leftrightarrow \exists y : |y| \leq p(|x|) \wedge M(x, y) \text{ accepte.}$$

(*) 1. Rappelez brièvement comment on prouve la direction « si » du Thm. 1, ainsi que la direction « seulement si ».

Solution:

« si » : Soit la machine à témoin M telle que « $x \in L \Leftrightarrow \exists y : M(x, y) \ldots$ ». Alors une machine non déterministe « classique » peut reconnaître L en devinant pour chaque x le y de taille $\leq p(|x|)$, en le stockant sur une bande de travail, puis en simulant M pour vérifier qu'on a bien $M(x, y)$. Cette machine classique est en temps polynomial comme M, et donc $L \in$ NP.

Réponse lapidaire attendue : « deviner y et vérifier $M(x, y)$ se fait dans NP. »

« seulement si » : Si $L \in$ NP alors il existe une machine non déterministe qui reconnaît L en temps polynomial $p(|x|)$. On peut supposer sans perte de généralité qu'à chaque étape cette machine a le choix entre exactement deux règles de transition. On transforme cette machine en machine à témoin M où le témoin y est une suite de $p(n)$ booléens utilisés à chaque étape de calcul comme une instruction disant s'il faut choisir la 1ère option non déterministe ou la 2ème option. Cette machine à témoin est déterministe et elle convient.

Réponse lapidaire attendue : « le témoin y indique pour chaque étape de calcul les choix non déterministes que doit effectuer la machine NP afin d'accepter $x \in L$. »

(*) 2. Que faut-il changer à votre preuve de la question 1 si on renforce le Thm. 1 en exigeant que la 2ème bande d'entrée de la machine à témoin soit à lecture unique (*read-once* en anglais), c.-à-d. que la tête de lecture sur y n'a pas le droit de se déplacer à gauche (de revenir en arrière) ?

Solution:
La direction « si » reste a fortiori valide. Pour la direction « seulement si » on note que les $p(n)$ booléens sur y sont lus une fois chacun et dans l'ordre fourni.

(**) 3. Dans une autre direction, que faut-il changer à votre preuve de la question 1 si on renforce le Thm. 1 en exigeant que la machine à témoin soit en espace logarithmique (et toujours déterministe) ?

Solution:
La direction « si » reste a fortiori valide puisqu'une machine en espace logarithmique est forcément en temps polynomial. Pour la direction « seulement si » la machine à témoin ne peut plus simuler en espace logarithmique la machine de départ munie d'une liste y des choix car les configurations (essentiellement la mémoire de travail) sont trop grandes. Mais on peut enrichir le témoin : au lieu de considérer un témoin constitué de $p(n)$ booléens, on prend un témoin qui liste les $p(n)$ configurations visitées lors d'un calcul acceptant de x. Maintenant les configurations sont données dans le certificat et les vérifications à faire sont moins gourmandes en mémoire : essentiellement, on vérifie bien que chaque configuration peut découler de la précédente par l'application d'une des règles de transition. Il suffit de lire une fenêtre de 2 ou 3 symboles sur une configuration pour déduire ce que peut être le symbole correspondant sur la configuration suivante : la vérification est en espace logarithmique.

(***) 4. Quelle classe de langages est obtenue si maintenant on combine les deux restrictions des questions précédents (machine déterministe logspace et témoin *read-once*) ? Justifiez votre réponse.

Solution:
On a maintenant $L \in \mathsf{NL}$.

Pour « si », la machine à témoin attend comme en question 1 un certificat y constitué de $p(n)$ booléens guidant les choix non. Il s'agit maintenant de simuler une machine logspace non déterministe, ce qu'une machine à témoin déterministe peut faire en logspace si on lui fournit y (qui comme en question 2 ne sera lu qu'une fois).

Pour « seulement si » on suppose qu'une machine logspace à témoin *read-once* reconnaît L : si au lieu de lire y on devine ses caractères un à un, on obtient une machine non déterministe en logspace, donc L est dans NL. Notons que cette construction marche car y n'est bien lu qu'une fois.

Problème 2 : Satisfaisabilité et insatisfaisabilité

On considère des formules booléennes $\varphi, \psi, \ldots$ construites comme d'habitude avec les connecteurs logiques $\wedge$, $\vee$, $\neg$, les constantes $\bot$ (faux) et $\top$ (vrai), et des variables booléennes $x_1, x_2, x_3, \ldots$ Ces formules ne sont pas forcément en forme clausale sauf si c'est explicitement précisé. On note $\mathit{Var}(\varphi)$ l'ensemble des variables qui apparaissent dans φ.

On s'intéresse aux problèmes de décision suivants :

SAT : l'ensemble des formules satisfaisables (ou plus exactement le langage des formules satisfaisables dans une notation appropriée) ;

UNSAT : l'ensemble des formules insatisfaisables (*unsatisfiable* en anglais), c.-à-d. des formules φ telles que $v(\varphi) = \bot$ pour toute valuation v des variables ;

SATUNSAT : l'ensemble des couples (φ, ψ) de formules telles que φ est satisfaisable et ψ ne l'est pas ;

SINGLESAT : l'ensemble des couples (φ, ψ) telles qu'une et une seule des deux formules est satisfaisable ;

PAIRSAT : l'ensemble des couples (φ, ψ) de deux formules satisfaisables ;

MINUNSAT : l'ensemble des formules φ en forme clausale $\varphi \equiv C_1 \wedge \cdots \wedge C_m$ telles que φ n'est pas satisfaisable mais le devient dès qu'on retire une clause quelconque (c.-à-d. $\varphi \in$ UNSAT et $\bigwedge_{\substack{1 \leq i \leq m \\ i \neq j}} C_i \in$ SAT pour tout $j = 1, \ldots, m$).

Enfin, pour deux problèmes $L \subseteq A^*$ et $L' \subseteq B^*$ on note $L \leq L'$ quand il existe une réduction de L dans L' (ici, et comme en cours, le mot « réduction » est un raccourci pour « réduction logspace »). Si maintenant $L \leq L' \leq L$ on dit que les deux problèmes sont équivalents.

(*) 5. Donnez une réduction montrant PAIRSAT $\leq$ SAT.

Solution:
$r(\varphi, \psi) \stackrel{\text{def}}{=} \varphi \wedge \psi'$ convient, où ψ' est obtenue à partir de ψ en renommant toutes ses variables de façon à garantir $\mathit{Var}(\varphi) \cap \mathit{Var}(\psi) = \emptyset$ (preuve de correction omise). Ce renommage peut évidemment être effectué en logspace, par exemple en commençant par identifier le plus grand indice k tel que x_k apparaît dans (φ, ψ) puis en remplaçant chaque x_i de ψ par x_{i+k}.

(*) 6. Donnez une réduction montrant SATUNSAT $\leq$ SINGLESAT.

Solution:
$r(\varphi, \psi) \stackrel{\text{def}}{=} (\varphi, \varphi \wedge \psi')$ convient (preuve de correction omise).

(*) 7. Donnez une réduction montrant SINGLESAT $\leq$ SATUNSAT.

Solution:
$r(\varphi, \psi) \stackrel{\text{def}}{=} (\varphi \vee \psi', \varphi \wedge \psi')$ convient (preuve de correction omise).

(*) 8. Montrez MINUNSAT $\leq$ SINGLESAT.

Solution:
Pour $\varphi \equiv \bigwedge_i C_i$ en forme clausale on notera $\varphi^{(-j)}$ la formule réduite $\bigwedge_{i \neq j} C_i$. On montre alors MINUNSAT $\leq$ SINGLESAT par la réduction

$$r(\varphi) = r\Big(\bigwedge_i C_i\Big) \stackrel{\text{def}}{=} \Big(\bigwedge_j \varphi^{(-j)\prime}, \varphi\Big)$$

où dans chaque $\varphi^{(-j)}$ les variables sont renommées de sorte que ces formules utilisent des ensembles de variables deux à deux disjoints.

Les questions suivantes cherchent à exhiber une réduction $\mathsf{SATUNSAT} \leq \mathsf{MINUNSAT}$ dans l'autre direction afin de pouvoir conclure $\mathsf{SATUNSAT} \equiv \mathsf{MINUNSAT}$ (ce qu'on fera à la question 13).

Pour ce faire, on considère un ensemble $Z = \{z_1, \ldots, z_n\}$ de n variables booléennes sur lesquelles on définit les clauses suivantes pour tous $i, j = 1, \ldots, n$ tels que $i \neq j$:

$$C_i^Z \equiv z_1 \vee \cdots \vee z_{i-1} \vee \neg z_i \vee z_{i+1} \vee \cdots \vee z_n, \qquad D_{i,j}^Z \equiv \neg z_i \vee \neg z_j.$$

On définit aussi une dernière clause C^Z, ainsi que la formule CNF φ^Z :

$$C^Z \equiv z_1 \vee z_2 \vee \cdots \vee z_n, \qquad \varphi^Z \equiv C^Z \wedge \bigwedge_{i=1}^{n} C_i^Z \wedge \bigwedge_{1 \leq i < j \leq n} D_{i,j}^Z.$$

(**) 9. Montrez que, quelque soit $n \in \mathbb{N}$, φ^Z est dans $\mathsf{MINUNSAT}$. (N'oubliez pas le cas $n = 0$.)

Solution:

Notons d'abord que φ^Z n'est pas satisfaisable : si une valuation v ne valide aucune des variables booléennes, elle ne valide pas la clause C^Z ; si v ne valide que la variable z_i, elle ne valide pas la clause C_i^Z correspondante ; si v valide deux variables ou plus, elle ne valide pas la clause $D_{i,j}^Z$ correspondante.

On voit maintenant comment φ^Z devient satisfaisable si on lui retire une clause quelconque. P.ex., si on retire C_i^Z, la valuation qui ne valide que z_i convient, et si on retire $D_{i,j}^Z$, c'est la valuation qui ne valide que z_i et z_j qui convient. Ainsi φ^Z est dans $\mathsf{MINUNSAT}$.

NB : Le raisonnement ci-dessus convient même si $n = 0$: on a alors $\varphi^Z = C^Z = \bigvee \emptyset = \bot$, c.-à-d. que φ^Z ne contient que C^Z, la clause vide qui est insatisfaisable. Si on retire cette clause, il subsiste $\bigwedge \emptyset = \top$, la conjonction vide qui est satisfaisable.

(***) 10. Montrez $\mathsf{UNSAT} \leq \mathsf{MINUNSAT}$. Pour cette réduction $\varphi \mapsto \varphi'$ on pourra supposer que φ est en CNF et, en s'inspirant de ce qui est fait à la question 9, que φ' est obtenue en ajoutant des littéraux à chaque clause de φ ainsi que des clauses supplémentaires qui contraignent les variables nouvelles.

Solution:

On suppose que $\varphi = F_1 \wedge \cdots \wedge F_n$ est une $\mathsf{3CNF}$ telle que $F_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ où $\ell_{i,j}$ est un littéral $x_{i,j}$ ou $\neg x_{i,j}$. (Notons qu'on peut supposer que les variables $x_{i,j}$ dans une clause F_i sont toutes distinctes : si F_i est de la forme $x \vee \neg x \vee \ell_{i,3}$ alors la clause est toujours validée et on peut la retirer de φ.)

Partant de φ, on construit de nouvelles clauses $F_i' \equiv F_i \vee C_i'$ où $C_i' \equiv z_1 \vee \cdots \vee z_{i-1} \vee \neg z_i \vee z_{i+1} \vee \cdots \vee z_n$, inspirée de la question 9, utilise des variables booléennes tirées de $Z = \{z_1, \ldots, z_n\}$ et distinctes de celles de φ. On définit aussi $G_{i,k}$ pour $k = 1, \ldots, 3$, via $G_{i,k} \equiv \neg \ell_{i,k} \vee C_i'$, ainsi que toutes les clauses $D_{i,j}^Z \equiv \neg z_i \vee \neg z_j$ comme dans la question 9.

Définissons alors $\varphi' \stackrel{\text{def}}{=} \bigwedge_i F_i' \wedge \bigwedge_i \bigwedge_{k=1}^{3} G_{i,k} \wedge \bigwedge_{i \neq j} D_{i,j}^Z$ et montrons que φ' est dans $\mathsf{MINUNSAT}$ ssi φ est insatisfaisable.

On a (i) si φ est satisfaisable, φ' l'est aussi : ici on étend la valuation v qui valide φ en posant $v(z_i) = \bot$ pour tout i ; (ii) si φ' est satisfaisable, φ l'est aussi : la valuation v validant φ' ne peut pas valider deux z_i distincts à cause des clauses $D_{i,j}^Z$, et si elle ne valide qu'un unique z_i, elle ne valide pas C_i' et donc doit valider la clause F_i correspondante —afin de valider F_i'— ainsi que les trois $\neg \ell_{i,k}$ —afin de valider chaque $G_{i,k}$— ce qui est contradictoire. Donc forcément v ne valide aucun des z_i et, puisqu'elle valide les clauses F_i', elle valide aussi les F_i et donc φ ; (iii) φ' devient satisfaisable dès qu'on lui retire une clause : pour chaque résidu il suffit de définir la valuation sur les z_i et le reste est indifférent (ou facile) : (iii/a) si on retire la clause $D_{j,j'}^Z$ il suffit d'invalider toutes les variables sauf z_j et $z_{j'}$ et ainsi on valide tous les C_i' donc les F_i' et les $G_{i,k}$, (iii/b) si on retire la clause F_j' il suffit de valider tous les z_i sauf z_j et d'invalider les trois $\ell_{i,k}$, (iii/c) si on retire la clause $G_{j,k'}$, il suffit de valider

les z_i sauf z_j et d'invalider les $\ell_{i,k}$ sauf $\ell_{i,k'}$ (afin que F'_j soit validée), En combinant (i–iii), on obtient $\varphi \in \mathsf{UNSAT}$ ssi $\varphi' \in \mathsf{MINUNSAT}$. Le fait d'avoir supposé φ en forme $\mathsf{3CNF}$ n'est pas une restriction car on sait transformer toute formule φ en une $\mathsf{3CNF}$ équivalente du point de vue de la satisfaisabilité (et donc aussi du point de vue de l'insatisfaisabilité mais, *attention !*, pas du point de vue de la validité).

(**) 11. Montrez que $\mathsf{SAT} \leq \mathsf{MINUNSAT}$. Pour ce faire on pourra continuer la construction des deux questions précédentes.

Solution:
On associe à φ la conjonction $\varphi'' \stackrel{\text{def}}{=} \varphi' \wedge C^Z$ avec φ' tiré de la question précédente et C^Z de la question 9. Maintenant φ'' n'est pas satisfaisable même si φ l'est. De plus, si φ n'est pas satisfaisable, on obtient la formule φ' (insatisfaisable comme φ) en retirant la clause C^Z de φ''. Et si φ est satisfaisable alors en retirant n'importe quelle clause de φ'' on obtient soit φ' qui est satisfaisable, soit une conjonction $C^Z \wedge \varphi'''$ où φ''' est un résidu de φ'' satisfaisable par une valuation qui n'est pas uniformément $\bot$ et qui donc valide aussi la clause C^Z. Au final, $\varphi'' \in \mathsf{MINUNSAT}$ ssi $\varphi \in \mathsf{SAT}$.

La suite du problème est indépendante des constructions des questions 9 à 11 dont on pourra admettre les conclusions.
On définit $\mathsf{TwoMINUNSAT} \stackrel{\text{def}}{=} \{(\varphi, \varphi') \mid \varphi, \varphi' \in \mathsf{MINUNSAT}\}$.

(**) 12. Donnez une réduction montrant que $\mathsf{TwoMINUNSAT} \leq \mathsf{MINUNSAT}$.

Solution:
À une instance (φ, φ') composée de deux formules booléennes en forme clausale $\varphi = \bigwedge_i C_i$ et $\varphi' = \bigwedge_j C'_j$ notre réduction associe $\psi \stackrel{\text{def}}{=} \bigwedge_i \bigwedge_j (C_i \vee C'_j)$. Ici on fait l'hypothèse que φ et φ' utilisent des ensembles de variables disjoints (et sinon la réduction renomme les variables de φ' p.ex.). Cette réduction est logspace et il s'agit maintenant d'en prouver la correction.
1. Puisque ψ n'est rien d'autre que la formule $\varphi \vee \varphi'$ mise sous forme clausale, on sait que ψ est satisfaisable ssi φ ou φ' l'est. Autrement dit, ψ est insatisfaisable ssi φ et φ' le sont.
2. Montrons maintenant que si φ, φ' sont dans $\mathsf{MINUNSAT}$ alors ψ l'est aussi. L'hypothèse nous donne déjà (point 1) que ψ n'est pas satisfaisable. Considérons un résidu $\psi \setminus (C_i \vee C'_j)$ quelconque. Puisque $\varphi \in \mathsf{MINUNSAT}$, il existe une valuation v qui valide $\varphi \setminus C_i$. Et il existe de même une valuation v' qui valide $\varphi' \setminus C'_j$. Dès lors $v \cup v'$ valide toutes les clauses $C_k \vee C'_\ell$ telles que $k \neq i \vee \ell \neq j$, et donc chaque résidu $\psi \setminus (C_i \vee C'_j)$ est satisfaisable.
3. Montrons enfin que si ψ est dans $\mathsf{MINUNSAT}$ alors φ, φ' le sont aussi. On suppose $\psi \in \mathsf{MINUNSAT}$ (donc φ et φ' ne sont pas satisfaisables et en particulier ne sont pas des conjonctions vides) et on considère un résidu $\varphi \setminus C_i$ quelconque de φ. On sait que, pour n'importe quel j, $\psi \setminus (C_i \vee C'_j)$ est satisfaisable, p.ex. par v. Puisque φ' n'est pas satisfaisable, une des clauses C'_k n'est pas validée par v. Néanmoins chaque $C_{i'} \vee C'_k$, pour $i' \neq i$, figure dans $\psi \setminus (C_i \vee C'_j)$ et donc est validée par v. On en déduit que v valide $C_{i'}$ pour tout $i' \neq i$. Donc il valide $\varphi \setminus C_i$ qui est donc satisfaisable. Ce raisonnement s'applique à tout résidu $\varphi \setminus C_i$ et on conclut $\varphi \in \mathsf{MINUNSAT}$. Il s'applique aussi à φ', montrant $\varphi' \in \mathsf{MINUNSAT}$. Notre réduction $(\varphi, \varphi') \mapsto \psi$ est donc correcte.

(*) 13. Montrez que $\mathsf{MINUNSAT} \equiv \mathsf{SATUNSAT}$.

Solution:
En combinant les réductions des questions 10 et 11, on obtient $\mathsf{SATUNSAT} \leq \mathsf{TwoMINUNSAT}$ et on déduit $\mathsf{SATUNSAT} \leq \mathsf{MINUNSAT}$ avec la question 12. Par ailleurs on a montré $\mathsf{MINUNSAT} \leq \mathsf{SATUNSAT}$ avec les questions 7 et 8.

On définit DP (pour « différence de langages NP ») via

$$L \in \mathsf{DP} \Leftrightarrow \exists L_1, L_2 \in \mathsf{NP} : L = L_1 \setminus L_2 \,.$$

(**) 14. Montrez qu'il existe des problèmes DP-complets.

Solution:
SATUNSAT est un bon candidat.
(1) Tout d'abord SATUNSAT est dans DP car il est facilement exprimé sous la forme $L_1 \setminus L_2$ avec $L_1, L_2 \in \mathsf{NP}$ définis comme des variantes du langage SAT.
(2) Maintenant soient L_1, L_2 des langages de NP et r_1, r_2 les réductions de ces langages dans SAT (qui existent car SAT est NP-complet). Alors (r_1, r_2) est une réduction de $L_1 \setminus L_2$ dans SATUNSAT.