TD8 - Classes de complexités

Exercice 1

Soient P_1 et P_2 deux problèmes de décision. Supposons connue une réduction polynomiale de P_1 en P_2. Répondre brièvement aux questions suivantes.

Question 1. Si $P_1 \in$ P, a-t-on $P_2 \in$ P ?

Question 2. Si $P_2 \in P$, a-t-on $P_1 \in$ P ?

Question 3. Si P_1 est NP-complet, P_2 est-il NP-complet ?

Question 4. Si P_2 est NP-complet, P_1 est-il NP-complet ?

Question 5. Si on connaît une réduction polynomiale de P_2 en P_1, P_1 et P_2 sont-ils NP-complets ?

Question 6. Si P_1 et P_2 sont NP-complets, existe-t-il une transformation polynomiale de P_2 en P_1 ?

Question 7. Si $P_1 \in$ NP, P_2 est-il NP-complet ?

Exercice 2

Montrer que le problème Path défini ci-dessous est dans P.

$$\textsc{Path} = \left\{ \langle G, s, t \rangle, \quad \begin{array}{l} G \text{ est un graphe, } s \text{ et } t \text{ sont deux sommets de } G \\ \text{et il existe un chemin entre } s \text{ et } t \text{ dans } G \end{array} \right\}$$

Exercice 3

Question 1. Étant donné un graphe G, rappeler la définition d'une *clique* et d'une *k-clique*

Question 2. Montrer que le problème Clique défini ci-dessous est dans NP.

$$\textsc{Clique} = \{\langle G, k \rangle, G \text{ est un graphe qui admet une } k\text{-clique}\}$$

Exercice 4

Montrer que le problème SubsetSum défini ci-dessous est dans NP.

$$\textsc{SubsetSum} = \left\{ \langle S, t \rangle, \quad \begin{array}{l} S \text{ est un ensemble fini d'entiers, } t \text{ est un entier} \\ \text{et il existe un sous-ensemble de } S \text{ dont la somme des éléments est } t \end{array} \right\}$$

Exercice 5

Une *couverture par sommets* (*vertex cover* en anglais) d'un graphe G est un ensemble C de sommets tel que chaque arête de $G = (V, E)$ est incidente à au moins un sommet de C. C'est donc un sous-ensemble de sommets $C \subseteq V$ tel que pour chaque arête (u, v) de G on a $u \in C$ ou $v \in C$. On dit que l'ensemble C couvre les arêtes de G.

Montrer que le problème VertexCover défini ci-dessous est dans NP.

$$\textsc{VertexCover} = \left\{ \langle G, k \rangle, \quad \begin{array}{l} G \text{ est un graphe, } k \text{ est un entier positif} \\ \text{et il existe un sous-ensemble de } S \text{ dont la somme des éléments est } t \end{array} \right\}$$

Exercice 6

Problème 3SAT

Le *problème 3SAT* est la restriction du problème SAT aux formules propositionnelles sous forme normale conjonctive dans lesquelles chaque clause contient au plus trois littéraux. En un sens, le problème 3SAT est plus simple que le problème SAT puisque ses formules ont une forme beaucoup mieux maîtrisée. Cependant, on peut démontrer que le problème reste NP-complet. On réalise ceci en montrant que toute formule propositionnelle φ peut être transformée en une formule φ' respectant les contraintes 3SAT, sans explosion de taille, et qui est satisfiable si et seulement si φ l'est également. La construction est basée sur deux éléments, formant une technique de transformation de formules appelée *transformation de Tseitin*.
Considérons pour commencer une formule $x \leftrightarrow (y \wedge z)$ avec trois variables propositionnelles x, y et z, et décomposons-la.

$$\begin{array}{lll} & x \leftrightarrow (y \wedge z) & \\ \equiv & (x \rightarrow (y \wedge z)) \wedge ((y \wedge z) \rightarrow x) & \text{décomposition de } \leftrightarrow \\ \equiv & (\neg x \vee (y \wedge z)) \wedge (\neg(y \wedge z) \vee x) & \text{décompositions de } \rightarrow \\ \equiv & (\neg x \vee y) \wedge (\neg x \vee y) \wedge (\neg(y \wedge z) \vee x) & \text{distributivité de } \vee \text{ sur } \wedge \\ \equiv & (\neg x \vee y) \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z \vee x) & \text{loi de de Morgan} \end{array}$$

Nous obtenons la conjonction de trois clauses avec au maximum trois littéraux chacune, c'est-à-dire une formule répondant à la restriction 3SAT. On peut vérifier de même les deux autres équivalences suivantes.

$$\begin{array}{rcl} x \leftrightarrow (y \vee z) & \equiv & (\neg x \vee y \vee z) \wedge (\neg y \vee x) \wedge (\neg z \vee x) \\ x \leftrightarrow \neg y & \equiv & (\neg x \vee \neg y) \wedge (y \vee x) \end{array}$$

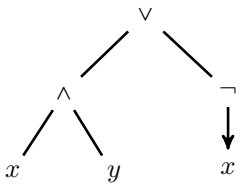
On peut donc mettre sous forme 3SAT toute formule obtenue par conjonction de formules ayant l'une des trois formes suivantes.

$$\begin{cases} x \leftrightarrow (y \wedge z) \\ x \leftrightarrow (y \vee z) \\ x \leftrightarrow \neg y \end{cases}$$

Il suffit alors de montrer que toute formule propositionnelle construite avec les trois connecteurs $\wedge$, $\vee$, $\neg$ peut effectivement être mise sous une telle forme. Intuitivement, partant d'une formule propositionnelle φ, nous allons associer à chaque connecteur de φ, c'est-à-dire à chaque sous-arbre de l'arbre de syntaxe abstraite de φ, une nouvelle variable propositionnelle destinée à représenter la validité de ce sous-arbre. Chacune de ces nouvelles variables sera alors incluse dans une formule de l'une des trois formes précédentes, en fonction du connecteur correspondant.

Transformation de Tseitin

Considérons la formule propositionnelle $\varphi = (x \wedge y) \vee (\neg x)$. Son arbre de syntaxe est le suivant.



On associe à la racine de cet arbre une nouvelle variable z_1, à son fils gauche la variable z_2 et à son fils droit la variable z_3. En écrivant la formule définissant chaque nœud, on obtient la conjonction suivante, qui est équisatisfiable avec φ.

$$z_1 \wedge (z_1 \leftrightarrow z_2 \vee z_3) \wedge (z_2 \leftrightarrow x \wedge y) \wedge (z_3 \leftrightarrow \neg x)$$

Il ne reste plus qu'à transformer chaque élément de cette conjonction en une formule 3SAT.

$$\begin{array}{rcl} \varphi & \equiv & z_1 \wedge (\neg z_1 \vee z_2 \vee z_3) \wedge (\neg z_2 \vee z_1) \wedge (\neg z_3 \vee z_1) \\ & & \wedge (\neg z_2 \vee x) \wedge (\neg z_2 \vee y) \wedge (\neg x \vee \neg y \vee z_2) \\ & & \wedge (\neg z_3 \vee \neg x) \wedge (x \vee z_3) \end{array}$$

Dans la preuve du théorème ci-dessous qui est l'objet de la suite de l'exercice, les deux idées précédentes sont réunies dans la définition d'une unique fonction récursive de transformation de formule.

Le problème 3SAT est NP-complet.

Question 1. Justifier brièvement que 3SAT appartient à la classe NP.

On définit une fonction f qui prend en entrée une formule φ et renvoie une paire (x, φ') telle que φ' est en forme 3SAT avec au maximum $3|\varphi|$ clauses, et telle que la conjonction $x \wedge \varphi'$ est satisfiable si et seulement si φ l'est. On se donne pour cela les équations récursives suivantes, également réalisées par le programme donné ci-dessous. Dans chacune des équations suivantes, x est une nouvelle variable. En outre, on note systématiquement $f(\varphi_1) = (y_1, \varphi'_1)$ et $f(\varphi_2) = (y_2, \varphi'_2)$ les applications de f à des sous-formules.

$$\begin{aligned}
f(z) &= (z, \mathsf{V}) \\
f(\mathsf{V}) &= (x, x) \\
f(\mathsf{F}) &= (x, \neg x) \\
f(\neg\varphi_1) &= (x, (\neg x \vee \neg y_1) \wedge (y_1 \vee x) \wedge \varphi'_1) \\
f(\varphi_1 \wedge \varphi_2) &= (x, (\neg x \vee y_1) \wedge (\neg x \vee y_2) \wedge (\neg y_1 \vee \neg y_2 \vee x) \wedge \varphi'_1 \wedge \varphi'_2) \\
f(\varphi_1 \vee \varphi_2) &= (x, (\neg x \vee y_1 \vee y_2) \wedge (\neg y_1 \vee x) \wedge (\neg y_2 \vee x) \wedge \varphi'_1 \wedge \varphi'_2)
\end{aligned}$$

Le programme (transformation de Tseitin) ci-dessous s'applique à une formule propositionnelle `f` utilisant des variables x_1 à x_n et construit une formule 3SAT utilisant les mêmes variables, et d'autres introduites pour les besoins de la construction. La fonction interne `new_var` introduit à chaque appel un numéro de variable non encore utilisé. La fonction `varmax` a déjà été rencontrée dans d'autres exercices.

```
let tseitin f =
  let nv = ref (varmax f) in
  let new_var () = incr nv; !nv in
  let rec mk_clauses = function
    | Var z -> z, []
    | True  -> let x = new_var () in x, [[x]]
    | False -> let x = new_var () in x, [[-x]]
    | Not f -> let x = new_var () in
               let y, cls = mk_clauses f in
               x, [-x; -y] :: [x; y] :: cls
    | Bin (op, f1, f2) ->
       let x = new_var () in
       let y1, cls1 = mk_clauses f1 in
       let y2, cls2 = mk_clauses f2 in
       let cls = match op with
         | And     ->
            [-x; y1] :: [-x; y2] :: [x; -y1; -y2] :: cls1 @ cls2
         | Or      ->
            [-x; y1; y2] :: [x; -y1] :: [x; -y2] :: cls1 @ cls2
         | Implies ->
            [-x; -y1; y2] :: [x; y1] :: [x; -y2] :: cls1 @ cls2
       in
       x, cls
  in
  let x, cls = mk_clauses f in
  { kind = CNF; nbvars = !nv; clauses = [x] :: cls }
```

La forme normale conjonctive f' renvoyée est satisfiable si et seulement si la formule d'origine f l'est. De plus, toute valuation satisfiant f' est également une valuation satisfiant f. On peut même facilement restreindre la valuation pour f' aux variables de f : il suffit de ne conserver que les entrées des n premières variables.

Question 2. Justifier brièvement que la formule φ' par cette fonction est en forme 3SAT. Combien de clauses comporte-t-elle, au maximum ?

Question 3. Montrer que'une valuation v pour les variables de φ satisfait φ si et seulement si elle peut être étendue en une valuation v' satisfiant $x \wedge \varphi'$.

Question 4. Montrer alors l'équisatisfiabilité de φ'.

Question 5. En déduire que tout problème SAT se réduit polynomialement en un problème 3SAT. Conclure sur la classe de complexité du problème 3SAT.