# Complexité structurelle – Partiel, correction

## Exercice 1

$\Longrightarrow$ : le langage $L = \{n$ en binaire ; $1^n \in T\}$ est dans $\mathrm{E}^T$ donc dans E. Pour décider $T$ dans $P$, sur l'entrée $1^n$ il suffit d'écrire $n$ en binaire et de lancer l'algo E pour $L$.
$\Longleftarrow$ : dans E, les appels à l'oracle sont au plus de taille $2^{O(n)}$, donc on peut simuler l'algorithme P tout en restant dans E.

## Exercice 2

**Question 1** – Notons d'abord que pour tout $C \subseteq A$, tout mot $x$ vérifiant $\exists y \in C$ tel que $f(x,y) = x$ ou $f(y,x) = x$ est dans $A$. Cela montre l'inclusion $\supseteq$.

Nous construisons maintenant l'ensemble $C$. Soit le graphe complet orienté dont les sommets sont les mots $x$ de $A \cap \{0,1\}^n$, et l'arc entre $x$ et $y$ est dans la direction $x$ vers $y$ si $f(x,y) = y$ et $f(y,x) = y$. Sinon on oriente l'arc arbitrairement.

La somme des degrés sortant est égale à la somme des degrés entrant, qui est égale au nombre d'arcs, soit $k(k-1)/2$ où $k$ est le nombre de sommets. Ainsi, il existe un sommet de degré sortant $\geq (k-1)/2$. On appelle $y_1$ ce sommet et on considère le nouveau graphe dans lequel on a enlevé tous les fils de $y_1$. Ce nouveau graphe a au plus $(k+1)/2$ sommets ; on peut recommencer avec lui pour trouver un sommet $y_2$ de degré sortant $\geq ((k+1)/2 - 1)/2$. Et ainsi de suite.

À la fin, on obtient au plus $n$ tels $y_i$, et il ne reste plus qu'un sommet dans le graphe, que l'on prend également : ainsi, on a un ensemble $C$ d'au plus $n+1$ sommets $y_i$. Par construction, tout sommet $x$ du graphe initial est le fils d'un des $y_i$. Cela signifie que soit $[f(x,y_i) = x$ et $f(y_i,x) = x]$, soit $[f(x,y_i) = y_i$ et $f(y_i,x) = x]$ ou symétriquement. On a bien le résultat demandé.

**Question 2** – On donne comme conseil l'ensemble $C$, qui est de taille $O(n^2)$ (il contient au plus $n+1$ mots de taille $n$). L'algorithme fonctionne alors ainsi : sur l'entrée $x$, pour tout $y \in C$ on calcule $f(x,y)$ et $f(y,x)$. Si l'un des deux est $x$ alors on accepte, sinon on rejette.

**Question 3** – Soit $A$ p-sélectif *via* $f$, et tel que $\mathrm{SAT} \leq A$. Ainsi, il existe $g$ polynomiale telle que $x \in \mathrm{SAT} \iff g(x) \in A$. Nous allons donner un algorithme polynomial pour SAT, utilisant comme d'habitude son auto-réductibilité.
Entrée : $\phi(x_1, \ldots, x_k)$.

- Pour $i$ de 1 à $k$, faire
  - on calcule $f(g(\phi(x_i := 0)), g(\phi(x_i := 1)))$
  - si le résultat est $g(\phi(x_i := 0))$, alors $\phi := \phi(x_i := 0)$, sinon $\phi := \phi(x_i := 1)$.
- maintenant, $\phi$ n'a plus de variable ; on accepte si et seulement si $\phi$ est vraie.

Si $\phi \notin \mathrm{SAT}$, alors l'algorithme refuse toujours. Si $\phi \in \mathrm{SAT}$, l'algorithme trouve une affectation la satisfaisant.

## Problème

**Question 1** – Il suffit de montrer que $\mathrm{NP}^{\mathrm{DP}} \subseteq \Sigma_2$ et le résultat suit par récurrence. En d'autres termes, il suffit de montrer que $\mathrm{NP}^{\mathrm{DP}} \subseteq \mathrm{NP}^{\mathrm{NP}}$. Si l'oracle DP est $L = L_1 \setminus L_2$ avec $L_1$ et $L_2$ dans NP, alors on peut simuler l'appel à l'oracle $L$ par un appel à $L_1$ et un appel à $L_2$, ce qui conclut.

**Question 2** – En combinant les deux étapes, on obtient $\text{coNP} \subseteq \text{NP}/1 \Longrightarrow \Sigma_2 \subseteq \text{DP}$. On conclut grâce à la question 1.

**Question 3** – À partir d'une formule $\phi$, en ajoutant des nouvelles variables on obtient facilement une formule $\psi$ de taille plus grande qui est satisfaisable si et seulement si $\phi$ l'est. Ainsi, il suffit de fixer un majorant $k$ (polynomial en $n$) sur la taille des requêtes à l'oracle, et au lieu de demander $\phi$ à l'oracle SAT, on demande $\psi$ de taille $k$.

**Question 4** – On veut simuler dans NP/1 le fonctionnement d'une machine NP $M$ qui fait des appels à l'oracle $\text{SAT} \in \text{NP}$. Soit $N_1$ une machine NP qui décide SAT. Puisque $\text{coNP} \subseteq \text{NP}/1$, on a par ailleurs une machine NP/1 $N_2$ qui décide coSAT.

On simule $M^{\text{SAT}}$ par une machine NP/1 de la façon suivante. Par la question 3, on peut supposer que tous les appels à l'oracle ont même taille $k$. À chaque appel à l'oracle pour savoir si $x \in \text{SAT}$, on lance d'abord $N_1$ sur $x$. Si sur le chemin en cours, la réponse est *oui*, alors on sait que $x \in \text{SAT}$ et on peut continuer la simulation de $M^{\text{SAT}}$. Si la réponse est *non*, alors on lance $N_2$ sur $x$ avec le conseil correspondant aux mots de taille $k = |x|$. Si la réponse de $N_2$ sur le chemin courant est *non*, alors on rejette (on arrête ce chemin de calcul), sinon on sait que $x \notin \text{SAT}$ et on peut continuer la simulation de $M^{\text{SAT}}$.

La machine ainsi définie fonctionne avec un bit de conseil, et a un chemin acceptant si et seulement si $M^{\text{SAT}}$ en a un.

**Question 5** – On sait qu'il existe des langages $\Sigma_2$-complets (par exemple, $\text{SAT}_2$) ; ils sont de la forme

$$L = \{x/(\exists y)(\forall z)(x, y, z) \in A]\},$$

où $A \in \text{P}$ et où les quantifications portent sur des mots de taille polynomiale. Il existe alors une réduction évidente de $L$ au langage donné dans l'énoncé.

**Question 6** – Si on connaît le bit de conseil, dans NP on peut décider $L$. On peut maintenant faire varier $y$, et à chaque fois appeler notre oracle $L$ sur l'entrée $(x, y)$, afin de faire une recherche dichotomique pour trouver le $y'_{max}$ de l'énoncé : notre entrée est rejetée si et seulement si $y'_{max} < y$. Le nombre d'étapes est logarithmique en le nombre de $y'$ possibles, donc polynomial en $n$ (c'est la taille de $y'$). Tout cela se fait donc dans $\text{P}^{\text{NP}}$.

**Question 7** – On suppose d'abord que le conseil est 0, et on simule le calcul de la question 6 avec ce conseil. Cela nous donne un mot $y'_1$. On fait de même en supposant que le conseil est 1 : on obtient un mot $y'_2$. Il suffit maintenant de décider si $(\forall z)[(x, y'_1, z) \in A] \vee (\forall z)[(x, y'_2, z) \in A]$. C'est un problème coNP qui se résoud en un appel à l'oracle supplémentaire. Ainsi, $L \in \text{P}^{\text{NP}}$ et donc $\Sigma_2 \subseteq \text{P}^{\text{NP}}$ (par complétude de $L$) sous l'hypothèse $\text{coNP} \subseteq \text{NP}/1$.

**Question 8** – Il est facile de construire une telle machine dans $\text{P}^{\text{LexMaxSat}} \subseteq \Sigma_2$. Maintenant, la simulation de la question 4 permet de conclure.

**Question 9** – $A_1$ et $A_2$ sont clairement dans NP.

Si $\phi \in \text{LexMaxSat}$ : $\phi \in A_1$ par définition de $A_1$. De plus, $\phi \notin A_2$ : si l'une des deux simulations rejette, alors $\phi$ est rejetée ; sinon, l'une des deux renvoie le bon résultat $x_{max}$, l'autre un $x_2$ plus petit, et $x_{max} = \max(x_{max}, x_2)$ se termine par 1.

Si $\phi \notin \text{LexMaxSat}$ : si $\phi \in A_1$ alors l'une des deux simulations renvoie un mot $x_0$ se terminant par 1. Il ne peut s'agir de $x_{max}$, sinon $\phi \in \text{LexMaxSat}$. Ainsi, puisque $\phi$ est satisfaisable, l'autre simulation renvoie $x_{max}$ qui se termine par 0. Or, puisque $x_0 < x_{max}$, $\phi \in A_2$, et donc $\phi \notin A_1 \setminus A_2$. Évidemment, si $\phi \notin A_1$ alors on a également $\phi \notin A_1 \setminus A_2$.

Ainsi, $\text{LexMaxSat} \in \text{DP}$ ce qui conclut par complétude de LexMaxSat.