Examen de calculabilité et complexité

M1 informatique – 3h

Le 12 janvier 2010

La qualité de la rédaction et de la présentation sera prise en compte dans l'évaluation. Le barème est donné à titre indicatif.

Exercice 1 – Vrai ou faux ? *(4 points)*

Les affirmations suivantes sont-elles vraies ou fausses ? Ne justifiez pas vos réponses. On comptera $+\frac{1}{3}$ point par bonne réponse, $-\frac{1}{3}$ point par mauvaise réponse, et 0 pour une absence de réponse. Si le total est négatif, on donnera 0 à l'exercice.

1. L'union de deux ensembles récursifs est encore récursive.
2. Si A et B sont des langages indécidables, alors $A \cap B$ est indécidable.
3. Si A est un langage indécidable, alors ${}^{c}A$ est indécidable.
4. Le complémentaire d'un ensemble fini est décidable.
5. Tout ensemble récursivement énumérable est décidable.
6. Le complémentaire d'un ensemble récursivement énumérable est récursivement énumérable.
7. Il existe des langages A et B, où A est infini et B fini, tels que $A \leq_m B$.
8. Si $A \subseteq B$ et B décidable, alors A décidable.
9. Une réduction est une fonction injective.
10. On sait simuler une machine de Turing non déterministe par une machine de Turing déterministe avec une perte de temps quadratique.
11. Le problème SAT est décidable.
12. On peut décider le problème Clique en espace exponentiel.

Exercice 2 *(3 points)*

1. Soit A un langage indécidable. Le langage $B = \{n \ : \ \exists x \in A, |x| \geq n\}$, où n est un entier codé en binaire, est-il décidable ?
2. Soit A un langage récursivement énumérable. Montrer que si pour tout n, A contient exactement un mot de taille n, alors A est décidable.

Exercice 3 *(3 points)*

1. Décrire une machine de Turing déterministe M, fonctionnant en espace logarithmique, qui reconnaît le langage $A = \{a^i b^j a^{i+j} \ : \ i, j \geq 0\}$. (Attention, quand il s'agit de compter l'espace, le ruban d'entrée est en lecture seule)

   Note : on ne demande pas la description complète de la machine mais seulement son principe de fonctionnement détaillé.
2. Quelle est la complexité en temps de M ?
3. Donnez la plus petite classe de complexité vue en cours dans laquelle vous pouvez placer le langage A.

Exercice 4 *(4 points)*

On rappelle que ES (Ensemble Stable) est le problème NP-complet suivant :

- Donnée : un graphe non orienté $G = (V, E)$ et un entier k ;
- Question : existe-t-il un sous-ensemble de V stable de taille k (appelé aussi sous-ensemble “indépendant”, c'est-à-dire k sommets n'ayant aucune arête entre eux) ?

On définit les deux problèmes suivants

CS (Couverture par Sommets) :
- Donnée : un graphe non orienté $G = (V, E)$ et un entier k ;
- Question : existe-t-il un sous-ensemble de V de taille k couvrant toutes les arêtes de G (c'est-à-dire k sommets tels que chaque arête ait l'une de ses extrémités parmi eux) ?

ED (Ensemble Dominant) :
- Donnée : un graphe non orienté $G = (V, E)$ et un entier k ;
- Question : existe-t-il un sous-ensemble de V dominant de taille k (c'est-à-dire k sommets $u_1, \ldots, u_k$ tels que pour tout autre sommet v, il existe i tel que $(u_i, v) \in E$) ?

1. Montrer que le problème CS est NP-complet. Pour cela, on pourra s'intéresser au complémentaire d'un ensemble stable.
2. Montrer que le problème ED est NP-complet. On pourra donner une réduction de CS en construisant un nouveau graphe dont l'ensemble des sommets est $V \cup E$.

Exercice 5 *(6 points)*

Soit $\Sigma = \{0, 1\}$ un alphabet à deux symboles. Un langage A sur l'alphabet Σ est *unaire* si $A \subseteq \{0\}^*$ (c'est-à-dire que les mots de A ne contiennent que des zéros). Le but de ce problème est de montrer le résultat suivant :

S'il existe un problème unaire NP-difficile, alors P = NP.

Supposons A unaire et NP-difficile : nous allons montrer que SAT $\in$ P. Puisque A est NP-difficile, il existe une réduction polynomiale f de SAT à A.

Soit $\phi(x_1, \ldots, x_n)$ une instance de SAT : nous devons décider en temps polynomial si $\phi \in$ SAT.

1. Que dire de ϕ si $f(\phi) \notin \{0\}^*$?
2. Pour une formule $\psi(x_1, \ldots, x_n)$, on désigne par ψ_0 la formule $\psi_0(x_2, \ldots, x_n) = \psi(0, x_2, \ldots, x_n)$ et par ψ_1 la formule $\psi_1(x_2, \ldots, x_n) = \psi(1, x_2, \ldots, x_n)$.
   Montrer que $\phi \in$ SAT ssi [$\phi_0 \in$ SAT ou $\phi_1 \in$ SAT].
   Pour simplifier, pour toute formule ψ on suppose que la taille $|\psi_0|$ (respectivement $|\psi_1|$) du codage de ψ_0 (resp. ψ_1) est égale à la taille $|\psi|$ du codage de ψ.
3. Tout au long de l'algorithme pour SAT que l'on cherche à décrire, on maintiendra un ensemble de formules booléennes. On appellera Φ_i cet ensemble à l'étape $i \leq n$. Au départ, on a $\Phi_0 = \{\phi\}$.
   On définit alors Φ_{i+1} par la procédure suivante :
   - $\Phi_{i+1} \leftarrow \emptyset$
   - $I \leftarrow \emptyset$
   - Pour tout $\psi \in \Phi_i$ faire
     - $u \leftarrow f(\psi_0)$
     - Si $u \in \{0\}^*$ et $u \notin I$ alors
       - $I \leftarrow I \cup \{u\}$
       - $\Phi_{i+1} \leftarrow \Phi_{i+1} \cup \{\psi_0\}$
     - $u \leftarrow f(\psi_1)$
     - Si $u \in \{0\}^*$ et $u \notin I$ alors
       - $I \leftarrow I \cup \{u\}$
       - $\Phi_{i+1} \leftarrow \Phi_{i+1} \cup \{\psi_1\}$
   - Renvoyer Φ_{i+1}

   Remarquer qu'on élimine une variable à chaque fois : l'ensemble Φ_i contient des formules à $n - i$ variables. Il y aura ainsi n étapes.
   Pour tout i, montrer que $\phi \in$ SAT si et seulement s'il existe $\psi \in \Phi_i$ satisfaisable.
4. Montrer qu'il existe un polynôme $p(n)$ tel que pour tout i, $|\Phi_i| \leq p(|\phi|)$ (ce polynôme dépend de la réduction f).
5. Conclure que SAT $\in$ P.