

# Examen de calculabilité et complexité

M1 informatique – 3h

Le 6 janvier 2012

Les notes de cours et de td sont autorisées. 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  $+\frac{1}{2}$  point par bonne réponse,  $-\frac{1}{2}$  point par mauvaise réponse, et 0 pour une absence de réponse. Si le total est négatif, on donnera 0 à l'exercice.

1. Il existe un langage dans  $\text{DTIME}(2^{8n})$  qui n'est pas dans  $\text{DTIME}(2^{3n})$ .
2. Tout langage est L-difficile pour les réductions  $\leq_m^p$ .
3. Si  $A$  est EXP-difficile (pour les réduction  $\leq_m^p$ ) alors  $A \notin \text{P}$ .
4. Si  $\text{SAT} \leq_m^p A$  et  $A \leq_m^p \text{SAT}$ , alors  $A$  est NP-complet.
5. Si  $A \in \text{PSPACE}$  alors  $A' = \{(x, 0^{2^{|x|}}) : x \in A\} \in \cup_k \text{DSPACE}((\log n)^k)$ .
6. Il existe un langage  $A$  tel que  ${}^c A \leq_m A$ .
7. Il existe un langage  $A$  tel que  ${}^c A \not\leq_m A$ .
8. Le langage  $\{\langle M \rangle : \exists x \in \Sigma^* \text{ tq } M(x) \text{ accepte}\}$  est décidable.

## Exercice 2

(3 points)

$M$  désigne une machine de Turing déterministe à 3 rubans. Soit  $A$  le langage suivant :

$A = \{(\langle M \rangle, x) : \text{lors du calcul } M(x), \text{ la machine n'écrit rien sur le deuxième ruban}\}.$

1. Montrer que le théorème de Rice ne permet pas de conclure quant à l'indécidabilité de  $A$ .
2. Montrer que  $A$  est indécidable.

## Exercice 3

(3 points)

Décrire une machine montrant que le langage des mots qui sont des carrés sur un alphabet  $\Sigma$  (c'est-à-dire le langage  $A = \{uu : u \in \Sigma^*\}$ ) est dans la classe L.

## Exercice 4

(3 points)

Montrer que le problème Half-3SAT suivant est NP-complet :

- *entrée* : une formule  $\phi(x_1, \dots, x_n)$  en 3-CNF avec un nombre pair de clauses ;
- *question* : existe-t-il une assignation des variables rendant vraie exactement la moitié des clauses ?

**Exercice 5**

(4 points)

Si  $A$  et  $B$  sont deux langages sur l'alphabet  $\Sigma$ , on désigne par  $A \times B$  le langage sur l'alphabet  $\Sigma$  égal à  $\{(x, y) : x \in A \text{ et } y \in B\}$  (produit cartésien de  $A$  et  $B$ ).

1. Montrer que  $\text{SAT} \times \text{SAT} \leq_m^p \text{SAT}$ .

L'objectif de cet exercice est de construire un langage  $A$  tel que  $A \times A \not\leq_m A$ . *Attention, contrairement à la question 1, il s'agit ici de réductions many-one fonctionnant en temps quelconque.*

2. Montrer qu'un langage  $A$  vérifiant  $A \times A \not\leq_m A$  est indécidable.

Nous allons construire un langage  $A$  ayant la propriété voulue en choisissant pour chaque mot  $x \in \Sigma^*$  si on le met dans  $A$  ou non. Le processus de construction est donc progressif, mot après mot. Lorsqu'on a choisi de mettre un mot  $x$  dans  $A$  ou de le mettre hors de  $A$ , nous dirons que  $x$  est *traité*; lorsque le sort de  $x$  n'a pas encore été fixé, nous dirons donc que  $x$  n'est pas traité.

Une machine de Turing  $M$  sera considérée comme calculant une fonction  $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ , c'est-à-dire que  $M$  calcule une réduction potentielle prenant en entrée un couple  $(x, y)$  (instance de  $A \times A$ ) et renvoyant un mot  $z$  (instance de  $A$ ).

Lors de notre construction progressive de  $A$ , nous allons faire en sorte que chacune de ces fonctions  $f$  ne soit pas une réduction de  $A \times A$  à  $A$ . Pour cela, on considère les codes de machines dans l'ordre lexicographique  $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ .

3. Supposons que les mots  $x_1, \dots, x_k$  ont déjà été traités de sorte qu'aucune des fonctions calculées par  $\langle M_1 \rangle, \dots, \langle M_{i-1} \rangle$  ne soit une réduction de  $A \times A$  à  $A$ . Soient  $x$  et  $y$  deux mots non traités.

En fonction de  $z = M_i(x, y)$ , comment traiter  $x, y$  et  $z$  de sorte que  $M_i$  ne soit pas une réduction de  $A \times A$  à  $A$ ? En d'autres termes, expliquer comment choisir si  $x, y$  et  $z$  doivent être mis dans  $A$  ou non. (Attention, il se peut que  $z$  ait déjà été traité auparavant.)

4. Conclure en faisant un choix arbitraire pour les mots non traités au cours du processus.

**Exercice 6**

(3 points)

Montrer que la machine  $M$  suivante vérifie la propriété :

« si  $P = NP$  alors  $M$  décide SAT en temps polynomial  
sur toutes les entrées suffisamment grandes »

(c'est-à-dire que  $M$  fonctionne en temps polynomial et qu'il existe  $m$  tel que si  $|\phi| > m$  alors

$[M(\phi) \text{ accepte ssi } \phi \in \text{SAT}]$ ).

On admettra qu'on peut énumérer les codes des machines fonctionnant en temps polynomial  $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ .

Fonctionnement de  $M$  sur l'entrée  $\phi$  de taille  $n$  :

- $i \leftarrow 1$
- tant que  $i \leq n$  faire
  - simuler  $M_i(\psi)$  pour toute formule  $\psi$  de taille  $\leq \log n$ ;
  - si pour tout  $\psi$ ,  $M_i$  donne toujours une affectation valide lorsque  $\psi \in \text{SAT}$ , sortir de la boucle;
- $i \leftarrow i + 1$
- simuler  $M_i(\phi)$ , produisant un résultat  $a$ ;
- si  $\phi(a) = 1$  accepter, sinon rejeter.

*Remarque :* comme dans tout langage de programmation, la valeur de  $i$  après la boucle tant que est égale à la valeur de  $i$  au moment de la sortie de la boucle si on sort naturellement.