Université Bordeaux – Master Sciences et Technologies, Informatique. Session automne 2015

# Examen Automates et Complexité, 16 décembre 2015, 14h – 17h

Documents autorisés : notes de cours et de TD.
**La notation attachera une grande importance à la clarté et à la concision des justifications.**
Le barème est indicatif. Sauf mention contraire, les questions sont indépendantes.

**Exercice 1 — Applications directes du cours (5 points).** Répondez aux questions suivantes en **justifiant brièvement** vos réponses (toute réponse non justifiée vaut 0 points) :

1) Existe-t-il une réduction du problème Sat au problème de correspondance de Post ?

2) On considère le langage $L$ des codes des machines de Turing qui acceptent si et seulement si leur entrée représente un nombre premier. Le langage $L$ est-il dans la classe **NP** ?

3) Si $L$ est un langage décidable de mots, est-il vrai que tout langage $K$ tel que $K \subseteq L$ est aussi décidable ?

4) Soit $f : \Sigma^* \to \Sigma^*$ une fonction calculable. Est-il vrai que pour tout langage semi-décidable, $L \subseteq \Sigma^*$, le langage $f(L)$ est également semi-décidable ?

5) Soit $f : \Sigma^* \to \Sigma^*$ une fonction calculable. Est-il vrai que pour tout langage décidable, $L \subseteq \Sigma^*$, le langage $f(L)$ est également décidable ? ∎

**Exercice 2 — NP-Complétude (6 points).** Dans cet exercice, on suppose que $\mathbf{P} \neq \mathbf{NP}$. Pour chacun des problèmes suivants, dites si il est **NP**-complet ou dans **P**. Attention, les réponses doivent être **prouvées**. Si vous répondez qu'un problème est **NP**-complet vous devez prouver qu'il est dans **NP** *et* qu'il est **NP**-difficile. Si vous répondez qu'un problème est dans **P** vous devez donner un algorithme polynomial qui le résout.

*Remarque : toutes les réductions sont faciles, la principale difficulté est de trouver le bon problème à réduire.*

1) Chemin Quasi-Hamiltonien

**ENTRÉE** : Un graphe orienté $G$.
**QUESTION** : Le graphe $G$ possède-t-il un chemin qui visite chaque sommet au moins une fois et au plus deux fois ?

On rappelle que dans une formule propositionnelle, un *littéral* est une variable ou une négation de variable. Un littéral est *positif* si c'est une variable, *négatif* si c'est une négation de variable.

2) Sat Modifié

Une *formule 4-CNF* est une conjonction de clauses, chacune du type $(\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4)$, où chaque $\ell_i$ est un littéral. Par exemple, $(x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_5 \vee \neg x_7)$ est une formule 4-CNF.

**ENTRÉE** : Une formule 4-CNF.
**QUESTION** : Y a-t-il une affectation des variables pour laquelle chaque clause a un littéral vrai et un littéral faux ?

3) Horn-Sat

Une *clause de Horn* est une disjonction de littéraux qui contient au plus un littéral positif et un nombre arbitraire de littéraux négatifs. Par exemple, "$x$", "$\neg y \vee \neg x$" et "$\neg x \vee \neg y \vee z$" sont des clauses de Horn. Inversement, "$x \vee y \vee \neg z$" n'est **pas** une clause de Horn car elle contient deux littéraux positifs, $x$ et $y$.

**ENTRÉE** : Une formule $\varphi$ qui est une conjonction de clauses de Horn.
**QUESTION** : La formule $\varphi$ est-elle satisfiable ?

4) Couverture par ensemble

**ENTRÉE** : Une liste d'ensembles finis d'entiers $E_1, \ldots, E_k$ et un nombre entier $n$.
**QUESTION** : Existe-t-il un ensemble d'entiers $E$ à $n$ éléments tels que pour tout $i = 1, \ldots, k$, on a $E \cap E_i \neq \emptyset$ ? ∎

**Exercice 3 — Machines à Compteurs (6 points).** Dans cet exercice, on présente un nouveau modèle de calcul : les *machines à compteurs*. Soit $k \geqslant 1$ un entier, une machine à $k$ compteurs est un tuple $M = (Q, q_0, q_a, q_r, \delta)$ où :

- $Q$ est un ensemble fini d'états.
- $q_0 \in Q$ est l'état initial.
- $q_a \in Q$ est l'état final acceptant.
- $q_r \in Q$ est l'état final rejetant.
- $\delta : Q \times \{Z, \neg Z\}^k \to Q \times \{-1, 0, 1\}^k$ est une fonction de transition, où $Z$ est un symbole (qui va permettre de tester si un compteur vaut 0).

Une transition est donc de la forme

$$\delta(q, (t_1, \ldots, t_k)) = (q', (n_1, \ldots, n_k))$$

où $q, q' \in Q$, chaque $t_i$ vaut soit $Z$ soit $\neg Z$, et où chaque $n_i$ vaut $-1$, 0 ou 1. On impose de plus la condition suivante pour toute telle transition :

$$\text{pour chaque } i \in \{1, \ldots, k\}, \text{ si } t_i = Z, \text{ alors } n_i \neq -1. \qquad (\mathcal{C})$$

On va définir l'exécution d'une machine à compteurs $M$ quelconque sur une valeur d'entrée $m \in \mathbb{N}$ (remarquez que l'entrée d'une machine à compteurs est donc un entier). La machine manipule $k$ variables $c_1, \ldots, c_k$, les « compteurs » de $M$, contenant des entiers positifs ou nuls. Une transition peut tester si chaque compteur vaut 0 ou non, et agir sur chaque compteur (en le décrémentant, ou en le laissant inchangé, ou en l'incrémentant).

On appelle *configuration* de $M$ un tuple $(q, c_1, \ldots, c_k)$ où $q$ est un état dans $Q$ et les $c_i$ sont des entiers **positifs ou nuls**. Si $C = (q, c_1, \ldots, c_k)$ et $D = (q', d_1, \ldots, d_k)$ sont deux configurations de $M$, on dit que $M$ passe de $C$ à $D$ par $\delta$, noté $C \xrightarrow{\delta} D$, si $\delta(q, (t_1, \ldots, t_k)) = (q', (n_1, \ldots, n_k))$ avec

- pour chaque $i$, on a $t_i = Z$ si $c_i = 0$ et $t_i = \neg Z$ si $c_i \neq 0$.
- pour chaque $i$, on a $d_i = c_i + n_i$.

Par exemple, si $C = (q, 3, 0, 4, 0)$ et $\delta(q, (\neg Z, Z, \neg Z, Z)) = (q', (-1, 1, 1, 0))$, alors $C \xrightarrow{\delta} D$ avec $D = (q', 2, 1, 5, 0)$. Autrement dit, dans l'état $q$, la transition $\delta(q, (\neg Z, Z, \neg Z, Z)) = (q', (-1, 1, 1, 0))$ teste que $c_1$ et $c_3$ sont non nuls, que $c_2$ et $c_4$ sont nuls, puis enlève 1 à $c_1$, ajoute 1 à $c_2$ et à $c_3$, laisse $c_4$ inchangé, et change l'état à $q'$. La condition $(\mathcal{C})$ interdit de décrémenter un compteur nul, ce qui garantit que chaque compteur reste positif ou nul. L'exécution de $M$ sur l'entrée $m \in \mathbb{N}$ est maintenant définie de la façon suivante : c'est une suite de configurations $C_0, C_1, C_2, \ldots$ (possiblement finie ou infinie, la machine peut ne pas s'arrêter) telles que :

- on part de la configuration $C_0 = (q_0, m, \underbrace{0, \ldots, 0}_{k-1})$. C'est-à-dire qu'on commence dans l'état initial $q_0$ et avec les compteurs nuls, sauf le premier qui contient l'entrée $m$.
- pour tout $i$, $C_i \xrightarrow{\delta} C_{i+1}$.

L'exécution termine dans une configuration $C_n$ si celle-ci utilise un des deux états finaux ($q_a$ ou $q_r$). Dans ce cas l'exécution est acceptante si elle termine dans l'état $q_a$ et rejetante si elle termine dans l'état $q_r$. Le langage d'une machine à compteur $M$ est l'ensemble $L(M)$ des entiers $m \in \mathbb{N}$ tels que l'exécution de $M$ sur $m$ est acceptante.

**I : Exemples de Machines à Compteurs.**

Pour les questions 1 et 2, on demande à la fois une explication intuitive de la machine à compteur demandée et sa description précise (c'est-à-dire sa liste de transitions). Pour les questions suivantes, on demande seulement des explications.

1) Donner une machine à 1 compteur $M$ dont le langage $L(M)$ est l'ensemble des entiers naturels pairs.

On dit que la machine à $k$ compteurs $M$ calcule une fonction totale $f : \mathbb{N} \to \mathbb{N}$ lorsqu'elle s'arrête dans $q_a$ sur toute entrée $m \in \mathbb{N}$, avec $c_1 = f(m)$ lorsque la machine s'arrête. Autrement dit, quand la machine s'arrête, elle est dans l'état $q_a$ et la valeur du premier compteur est $f(m)$.

2) Donner une machine à deux compteurs qui calcule le quotient entier de son entrée par 2 : si $m = 2x$ ou $m = 2x + 1$ avec $x \in \mathbb{N}$, la machine calcule $x$.

3) Décrire une machine à deux compteurs qui calcule la fonction $f(m) = 2m$.

**II : Machines à Compteurs et Machines de Turing.**

4) Soit $k \geqslant 1$ quelconque. Montrer que toute machine à $k$ compteurs peut être simulée par une machine de Turing (à plusieurs bandes). En d'autres termes, on demande d'expliquer comment, à partir du code source d'une machine à $k$ compteurs, on peut construire une machine de Turing qui accepte le même langage.

5) Montrer que toute machine de Turing à 1 bande peut être simulée par une machine à 4 compteurs. En d'autres termes, on demande d'expliquer comment, à partir du code source d'une machine de Turing, on peut construire une machine à 4 compteurs qui accepte le même langage.

*Indication : on pourra s'inspirer du codage utilisé dans le DS2 pour coder une machine de Turing par une machine à piles.*

6) Montrer que le problème suivant est indécidable :

**ENTRÉE** : Une machine $M$ à $k \geqslant 4$ compteurs et $m \in \mathbb{N}$ une entrée pour $M$.
**QUESTION** : Est-ce que $m \in L(M)$ ?

■

**Exercice 4 — Formules Booléennes Quantifiées (5 points).** On suppose fixé un ensemble $\mathcal{V}$ de variables. Une formule booléenne quantifiée est définie par induction à partir des éléments suivants :

— *Valeurs de vérité* : "Vrai" et "Faux" sont des formules.

— *Variables* : pour toute variable $x \in \mathcal{V}$, "$x$" est une formule.

— *Connecteurs logiques* : si $\varphi$ et $\psi$ sont des formules, "$\varphi \vee \psi$", "$\varphi \wedge \psi$" et "$\neg\varphi$" sont des formules.

— *Quantifications* : si $\varphi$ est une formule et $x \in \mathcal{V}$ est une variable, alors "$\exists x\ \varphi$" et "$\forall x\ \varphi$" sont des formules.

Par exemple "$\forall x \exists y \exists z\ x \vee (\neg x \wedge y \wedge \neg z)$" est une formule booléenne quantifiée. On va se restreindre aux formules qui vérifient les propriétés suivantes :

— Nos formules n'ont *pas de variables libres* sauf dans la question 4 : si la formule contient la variable $x$, alors $x$ se trouve sous une quantification "$\exists x$" ou "$\forall x$".

— Nos formules *quantifient chaque variable une seule fois* : pour toute variable $x \in \mathcal{V}$, la formule contient une seule quantification "$\exists x$" ou "$\forall x$".

— Nos formules ont tous leurs quantifications en tête, c'est-à-dire sont de la forme $Q_1 x_1 Q_2 x_2 \dots Q_k x_k\, \varphi$ où chaque $Q_i$ est soit $\forall$, soit $\exists$, et $\varphi$ n'a pas de quantification.

La valeur de vérité d'une formule booléenne quantifiée est définie avec l'interprétation habituelle des connecteurs et des quantifications. Par exemple, la formule $\forall x\, \exists y\ (x \wedge y) \vee (\neg x \wedge \neg y)$ est vraie (pour $x$ Vrai, on choisit $y$ Vrai, et pour $x$ Faux, on choisit $y$ Faux). Par contre, la formule $\forall x\, \forall y\ (x \vee y)$ est fausse.

Le problème Formule Booléenne Quantifiée est le suivant :

**ENTRÉE** : Une formule Booléenne quantifiée $\varphi$.
**QUESTION** : Est-ce que $\varphi$ est vraie ?

1) Montrer que tout problème **NP** se réduit polynomialement à Formule Booléenne Quantifiée.

2) Montrer que le problème Formule Booléenne Quantifiée est dans la classe **PSPACE**. Autrement dit, il faut expliquer comment calculer la valeur d'une formule booléenne quantifiée en utilisant un espace polynomial, de façon déterministe.

Dans les questions 3 et 4, on considère un problème $A \in$ **PSPACE** et une machine $M_A$ qui résout $A$ et travaille en espace polynomial.

3) Évaluer la taille du graphe des configurations de $M_A$ en fonction de la taille $n$ de l'entrée.

4) (**Difficile**) On dit que deux configurations $C, D$ de $M_A$ ont la propriété $\mathcal{P}_n$ s'il existe un chemin de taille au maximum $2^n$ dans le graphe des configurations de $M_A$ allant de $C$ à $D$.

En représentant les configurations avec des variables (comme dans la preuve du théorème de Cook), expliquer comment écrire pour chaque entier $n$, une formule booléenne quantifiée $\varphi_n$ de taille polynomiale, ayant des variables libres, et qui est satisfiable si et seulement s'il existe deux configurations ayant la propriété $\mathcal{P}_n$ (commencer par $\varphi_0$ et calculer $\varphi_{n+1}$ en fonction de $\varphi_n$).

5) Utiliser les questions précédentes pour montrer que Formule Booléenne Quantifiée est **PSPACE**-complet.

6) Quelle relation entre **PSPACE** et **NPSPACE** peut-on déduire de la question 4 ? Justifier la réponse. ■

Fin