université de BORDEAUX	**Année Universitaire 2023–2024, Master CSI** Examen, Théorie de la complexité (UE 4TCY701U) 21 décembre 2023, 14h30–17h30	**Collège Sciences et Technologies**

La rédaction doit être précise et concise. Le barème est indicatif.
Seules, les réponses correctement justifiées apporteront des points.
Vous pouvez utiliser les résultats de questions antérieures, même non traitées.

Exercice 1 – Applications du cours (2 points)

Répondez aux questions suivantes en quelques lignes, en justifiant clairement vos réponses.

1. Soit $f : \Sigma^* \to \Sigma^*$ une fonction calculable. Est-il vrai que pour tout langage semi-décidable $L \subseteq \Sigma^*$, le langage $f(L)$ également semi-décidable ?
2. On considère le langage L des codes des machines de Turing qui acceptent si et seulement si leur entrée représente un nombre premier écrit en binaire. Le langage L est-il dans la classe **NP** ?

Exercice 2 – Complexité (5 points)

Pour chaque problème ci-dessous, donner la plus petite classe qui le contient parmi **L**, **NL**, **P**, **NP**, **PSPACE** en justifiant la réponse donnée. De plus, si vous répondez **NL**, **NP** ou **PSPACE**, déterminez s'il est complet pour cette classe.

Remarques

- Les réductions sont *simples* en choisissant le bon problème de départ.
- Pour les réductions, vous pouvez utiliser *tout* problème des notes de cours-TD ou du DS.
- Pour montrer qu'un problème est $\mathscr{C}$-complet, où $\mathscr{C}$ est une classe, pensez à vérifier qu'il est dans $\mathscr{C}$.

1. **Tautologie pour formules CNF**
   Entrée : Une formule propositionnelle en forme **CNF**.
   Question : La formule est-elle vraie pour **toute** affectation des variables ?

2. **Facteur de campagne**
   Entrée : Un graphe non dirigé $G = (S, A)$ où S est l'ensemble des sommets de G et A celui des arêtes de G, une fonction $\ell : A \to \mathbb{N}$, un sous-ensemble A' de A et un entier $K \geqslant 0$.
   Question : Y a-t-il un chemin fermé $s_0 - s_1 - \cdots - s_{n-1} - s_n = s_0$ dans G qui contient chaque arête de A' et tel que $\ell(s_0, s_1) + \ell(s_1, s_2) + \cdots + \ell(s_{n-1}, s_0) \leqslant K$?

Un *semigroupe* est un ensemble S muni d'une loi *associative* $(x, y) \mapsto xy$ (on a $(xy)z = x(yz)$ pour tous $x, y, z \in S$).

3. **Semigroupe commutatif**
   Entrée : Un semigroupe S donné par sa table de multiplication.
   Question : Le semigroupe S est-il commutatif, c'est-à-dire, a-t-on $xy = yx$ pour tous éléments x, y de S ?

Si S est un semigroupe et X un sous-ensemble de S, le *semigroupe engendré par* X est l'ensemble de tous les produits $x_1 x_2 \cdots x_n$ avec $n > 0$ et où chaque $x_i \in X$ (un même élément peut apparaître plusieurs fois dans ce produit).

4. **Appartenance à un sous-semigroupe**
   Entrée : Un semigroupe S donné par sa table de multiplication, un sous-ensemble $X \subseteq S$ et $x \in S$.
   Question : L'élément x appartient-il au semigroupe engendré par X ?

5. **Sac à dos**
   Entrée : Deux suites d'entiers $v_1, \ldots, v_n \in \mathbb{N}$ et $p_1, \ldots, p_n \in \mathbb{N}$ de la même longueur n et deux entiers $V, P \in \mathbb{N}$.
   Question : Existe-t-il un ensemble $I \subseteq \{1, \ldots, n\}$ tel que $\sum_{i \in I} p_i \leqslant P$ et $\sum_{i \in I} v_i \geqslant V$.

Exercice 3 – Machines reset (5 points)

On considère des machines de Turing « *reset* ». Celles-ci ont une seule bande dont les cases sont numérotées par les entiers naturels, la case la plus à gauche par 0. L'entrée est écrite à partir de la case 0, et initialement la tête est sur la case 0. On dispose de deux types d'instructions :

- Type 1 : la machine remplace le symbole sous la tête, change d'état, et déplace la tête d'une case à droite.
- Type 2 : la machine remplace le symbole sous la tête, change d'état, et repositionne la tête sur la case 0 (c'est-à-dire, sur la case la plus à gauche de la bande).

Par contre, la machine n'a pas d'instruction pour déplacer la tête d'une case vers la gauche. Une telle machine **peut être non déterministe**. Avec les notations vues en cours, on a donc une fonction de transition $\delta : Q \times \Gamma \to \mathscr{P}(Q \times \Gamma \times \{\triangleright, \lhd\lhd\})$, où $\triangleright$ désigne le déplacement de la tête d'une case à droite et $\lhd\lhd$ le repositionnement de la tête en début de bande (et où, si E est un ensemble, $\mathscr{P}(E)$ désigne l'ensemble des sous-ensembles de E). On considère des **machines de décision** : un calcul sur une entrée peut accepter, rejeter ou ne pas terminer.

1. Écrire une machine *reset* qui teste si un nombre binaire passé en entrée est multiple de 2.
2. Soit $\mathscr{M}$ une machine de Turing « classique » ($\mathscr{M}$ a des instructions pour déplacer la tête de lecture vers la gauche ou la droite et pour la laisser sur place, mais pas pour la ramener directement en position 0). Construire une machine *reset* $\mathscr{M}_{\text{reset}}$ qui simule $\mathscr{M}$: c'est-à-dire que pour tout mot d'entrée x, $\mathscr{M}$ doit accepter x si et seulement si $\mathscr{M}_{\text{reset}}$ accepte x.

Indication. Il faut expliquer comment simuler les instructions non-disponibles dans les machines *reset*. On dispose du non-déterminisme et on peut étendre l'alphabet pour « marquer » des cases pendant le calcul.

3. Le problème suivant est-il décidable ? Justifier précisément la réponse.
   ENTRÉE : Le code d'une machine *reset*.
   QUESTION : Cette machine *reset* s'arrête-t-elle sur le mot vide ?

On considère maintenant les machines *reset* déterministes. Une telle machine est donnée par une fonction de transition $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\triangleright, \lhd\lhd\}$.

4. Écrire une machine reset déterministe effectuant la division par 2 du nombre binaire donné en entrée.
5. Expliquer comment simuler une machine de Turing à une bande habituelle par une machine reset déterministe.
6. Montrer que si un problème est dans la classe **P**, il peut être décidé par une machine reset déterministe effectuant un nombre polynomial de pas de calcul par rapport à la taille de l'entrée.

Exercice 4 – Problèmes de décision sur les automates finis (12 points)

On considère des automates finis (déterministes ou non suivant les questions). Étant donné un alphabet fini Σ, on note $\mathscr{A} = (\Sigma, Q, I, F, \delta)$ un automate sur l'alphabet Σ, où,

- Q est l'ensemble des états de $\mathscr{A}$,
- $I \subseteq Q$ est l'ensemble des états initiaux de $\mathscr{A}$,
- $\delta \subseteq Q \times \Sigma \times Q$ est l'ensemble des transitions de $\mathscr{A}$,
- $F \subseteq Q$ est l'ensemble des états finaux de $\mathscr{A}$.

Comme d'habitude, on note une transition (p, a, q) de façon graphique, comme $p \xrightarrow{a} q$. Un automate peut donc être représenté par un graphe dirigé dont les arcs sont étiquetés par des lettres. Si $\mathscr{A}$ est un automate, on note $L(\mathscr{A})$ le langage des mots de Σ^* reconnus par $\mathscr{A}$.

On s'intéresse à des problèmes concernant les langages reconnus par des automates. Les questions demandent de déterminer dans quelles classes se trouvent certains problèmes (**en justifiant précisément chaque réponse**). Lorsqu'une question ne précise pas la classe, on demande de déterminer la plus petite classe vue en cours qui contient le problème. De plus, si la classe est **NL**, co-**NL**, **PSPACE**, **NP** ou co-**NP** et que le problème est complet pour cette classe, **on demande de l'indiquer et de le démontrer**.

Problèmes classiques

1. Quelle est la plus petite classe de complexité vue en cours contenant le problème suivant ? Justifier.

Test du vide pour un automate fini quelconque		
Entrée	:	Un alphabet Σ et un automate fini $\mathscr{A}$ **déterministe ou non** sur Σ.
Question	:	Le langage $L(\mathscr{A})$ est-il vide ?

2. Même question pour le problème suivant.

Appartenance d'un mot au langage d'un automate fini quelconque		
Entrée	:	Un alphabet Σ, un automate fini **déterministe ou non** $\mathscr{A}$ sur Σ et un mot $w \in \Sigma^*$.
Question	:	Le mot w appartient-il au langage $L(\mathscr{A})$?

3. On s'intéresse maintenant au problème suivant portant sur **deux** automates. Quelle est sa complexité ?

Intersection pour deux automates finis quelconques		
Entrée	:	Un alphabet Σ et deux automates finis **déterministes ou non** $\mathscr{A}_1$ et $\mathscr{A}_2$ sur Σ.
Question	:	Le langage $L(\mathscr{A}_1) \cap L(\mathscr{A}_2)$ est-il vide ?

Problème de l'alphabet commun

Étant donné un alphabet Σ et un mot $w \in \Sigma^*$, on note $\mathbf{alph}(w) \subseteq \Sigma$ l'ensemble des lettres contenues dans w. Par exemple, si $\Sigma = \{a,b,c,d\}$, on a $\mathbf{alph}(\varepsilon) = \emptyset$, $\mathbf{alph}(b\,c\,b\,b\,b\,c\,b) = \{b,c\}$ et $\mathbf{alph}(a\,a\,b\,b\,a\,d\,d\,b\,a) = \{a,b,d\}$.

Mots de même alphabet reconnus par deux automates quelconques		
Entrée	:	Un alphabet Σ et deux automates finis **déterministes ou non** $\mathscr{A}_1$ et $\mathscr{A}_2$ sur Σ.
Question	:	Existe-t-il $w_1 \in L(\mathscr{A}_1)$ et $w_2 \in L(\mathscr{A}_2)$ tels que $\mathbf{alph}(w_1) = \mathbf{alph}(w_2)$?

4. Montrer que ce problème est dans **NP**.
5. Montrer que ce problème est **NP**-difficile par réduction à partir de 3-**SAT**.

> Indication
>
> Étant donnée une conjonction de clauses $C_1 \wedge \cdots \wedge C_n$ utilisant les variables $x_1, \ldots, x_k$, on doit construire un alphabet Σ et deux automates $\mathscr{A}_1$ et $\mathscr{A}_2$. Intuitivement, les conditions suivantes doivent être satisfaites :
>
> - Pour tout mot $w \in \Sigma^*$ accepté par $\mathscr{A}_1$, $\mathbf{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$.
> - Pour tout mot $w \in \Sigma^*$ tel que $\mathbf{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$, si w est accepté par $\mathscr{A}_2$, alors l'affectation encodée satisfait toutes les clauses $C_1, \ldots, C_n$.

Problème de l'intersection d'un nombre arbitraire d'automates

Intersection d'un nombre arbitraire d'automates finis déterministes		
Entrée	:	Un alphabet Σ, un entier $n \geqslant 1$ et n automates finis **déterministes** $\mathscr{A}_1, \ldots, \mathscr{A}_n$ sur Σ.
Question	:	Le langage $\bigcap_{1 \leqslant i \leqslant n} L(\mathscr{A}_i)$ est-il vide ?

6. Montrer que ce problème est dans **PSPACE**.

On va montrer que le problème est **PSPACE**-difficile par réduction d'un **problème quelconque** dans **PSPACE**.

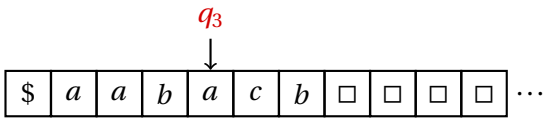
7. Écrivez l'objectif à réaliser, c'est-à-dire (1) ce qu'une telle réduction doit prendre en entrée, (2) ce qu'elle doit construire et (3) quelles sont les propriétés que cette réduction doit satisfaire.

On va utiliser une version modifiée du codage des configurations d'une machine de Turing à une bande par des mots vu plusieurs fois en cours (et en DS). On considère une machine $\mathscr{M} = (\Sigma, \Gamma, Q, q_0, q_a, q_r, \delta)$ dont l'espace est borné par une fonction polynôme $f : \mathbb{N} \to \mathbb{N}$ et on suppose que $Q \cap \Gamma = \emptyset$. Pour la définition, on fixe un mot d'entrée $w \in \Sigma^*$ pour $\mathscr{M}$: l'espace utilisé par $\mathscr{M}$ sur w est donc borné par l'entier $k = f(|w|)$. On code une configuration de l'exécution de $\mathscr{M}$ sur w par le mot $uqv \in (Q \cup \Gamma)^{k+1}$ de longueur $k+1$ où :

- q est l'état de contrôle dans lequel se trouve la machine pour cette configuration.
- le mot $uv \in \Gamma^k$ de longueur k est écrit sur sa bande (avec des blancs $\square$ dans toutes les cases non-utilisées),
- la tête de lecture pointe sur la première lettre qui suit u.

Par exemple, la configuration suivante est codée par le mot $\$aab\,q_3\,acb\,\square \cdots \square$ (avec suffisamment de lettres « $\square$ » pour que le mot soit de longueur $k+1$). Remarquez que la modification du codage par rapport à celui vu en cours est ce remplissage des mots de codage avec des $\square$ à la fin pour qu'ils aient tous la même taille $k+1 = f(|w|)+1$.

Bande de la machine :



On considère maintenant une nouvelle lettre $\# \notin Q \cup \Gamma$ et on note Λ le nouvel alphabet étendu $\Lambda = Q \cup \Gamma \cup \{\#\}$.

8. Montrer qu'il existe un algorithme en **temps polynomial** qui, étant donné un mot $w \in \Sigma^*$ pris en entrée, construit une séquence finie d'automates $\mathscr{A}_1, \ldots, \mathscr{A}_n$ sur l'alphabet Λ telle que :
   - Si le mot w est accepté par $\mathscr{M}$, alors $\bigcap_{1 \leqslant i \leqslant n} L(\mathscr{A}_i)$ est un **singleton** contenant uniquement le mot $C_0\#C_1\#C_2\#\ldots\#C_\ell\# \in \Lambda^*$, où $C_0 \vdash C_1 \vdash C_2 \vdash \cdots \vdash C_\ell$ est la séquence de configurations dans l'exécution acceptante de la machine $\mathscr{M}$ sur w.
   - Si w est rejeté par $\mathscr{M}$, alors $\bigcap_{1 \leqslant i \leqslant n} L(\mathscr{A}_i)$ est **vide**.

   > **Indication**. Intuitivement, chaque automate doit vérifier une propriété faisant que le mot d'entrée représente le calcul valide de la machine $\mathscr{M}$ sur le mot d'entrée. Écrivez l'ensemble de ces propriétés en français avant de concevoir les automates. Chacun d'eux est simple.

9. En déduire que le problème d'intersection d'automates finis déterministes est **PSPACE**-difficile.

Problèmes d'universalité et d'équivalence :

10. On considère l'universalité pour les automates *déterministes*. Quelle est la complexité du problème suivant ?

Universalité pour un automate fini déterministe		
Entrée	:	Un alphabet Σ et un automate fini **déterministe** $\mathscr{A}$ sur Σ.
Question	:	Le langage $L(\mathscr{A})$ est-il égal à Σ^* ?

11. Quelle est la complexité du même problème pour des automates qui ne sont pas nécessairement déterministes ?

Universalité pour un automate fini quelconque		
Entrée	:	Un alphabet Σ et un automate fini **déterministe ou non** $\mathscr{A}$ sur Σ.
Question	:	Le langage $L(\mathscr{A})$ est-il égal à Σ^* ?

12. On considère maintenant le problème d'équivalence. Quelle est la complexité du problème suivant ?

Équivalence de deux automates finis quelconques		
Entrée	:	Un alphabet Σ et deux automate finis **déterministes ou non** $\mathscr{A}_1$ et $\mathscr{A}_2$ sur Σ.
Question	:	Est-ce que $L(\mathscr{A}_1) = L(\mathscr{A}_2)$?