Année Universitaire 2018/2019, Session 1 d'automne Master CSI, UE 4TCY701U, Théorie de la complexité. Lundi 19 novembre 2018. 9h30–12h30 Durée : 3h. **Documents non autorisés.**	**Collège Sciences et Technologies**

Toute réponse non justifiée n'apportera pas de point. Le barème est indicatif.

Exercice 1 : Questions de cours — 3 Points

Répondez aux questions suivantes en **justifiant brièvement mais précisément** vos réponses (toute réponse non justifiée vaut 0 point) :

1) Le problème de l'arrêt (*HALT*) est-il dans **NP** ? **NP**-difficile ? **NP**-complet ?
2) On considère les machines de Turing qui utilisent un espace borné par $2^{|w|}$ où w est l'entrée de la machine. Le problème *HALT* restreint à ces machines est-il décidable ?
3) On considère un problème décidable P quelconque. Est-il vrai que tout problème décidable Q se réduit à P ? Ici, on considère des réductions calculables arbitraires (*i.e.*, pas forcément polynomiales).

Exercice 2 : Décidabilité ou indécidabilité — 3 Points

Étant donnés deux alphabets Σ_1, Σ_2, on étend toute fonction $f : \Sigma_1 \to (\Sigma_2)^*$ en une fonction de $(\Sigma_1)^*$ dans $(\Sigma_2)^*$, encore notée f, de la façon suivante : si $w = a_1 a_2 \cdots a_n \in (\Sigma_1)^*$, on définit $f(w) = f(a_1) f(a_2) \cdots f(a_n)$.

Par exemple, si $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{c, d, e\}$ et $f : \Sigma_1 \to (\Sigma_2)^*$ est définie par $f(a) = dde$ et $f(b) = dc$, on aura,

$$f(abba) = ddedcdcdde$$

Le problème ci-dessous est-il décidable ou indécidable ? On ***prouvera*** la réponse (par un algorithme ou une réduction).

ENTRÉE : Deux alphabets Σ_1, Σ_2 et deux fonctions $f : \Sigma_1 \to (\Sigma_2)^*$ et $g : \Sigma_1 \to (\Sigma_2)^*$.
QUESTION : Est-ce qu'il existe un mot non-vide $w \in (\Sigma_1)^*$ tel que $f(w) = g(w)$?

Exercice 3 : Un problème EXPTIME-complet — 3 Points

On considère le problème de décision suivant :

ENTRÉE : Une machine de Turing M, un mot d'entrée w et un entier $k \in \mathbb{N}$ écrit en base 2.
QUESTION : Est-ce que M accepte w après k transitions ou moins ?

1) Étant donnée une entrée M, w, k pour ce problème, quelle est la taille de cette entrée en mémoire ?
2) Montrer que ce problème est dans **EXPTIME**.
3) Montrer que ce problème est **EXPTIME**-difficile (et donc **EXPTIME**-complet).

Indication

La réduction est simple. Il s'agit plus de maîtriser les définitions et de savoir poser le problème que d'inventer quelque chose. En particulier, on remarquera que nous n'avons pas vu de problème **EXPTIME**-complet en cours : vous pouvez donc en déduire ce qu'il faut faire.

Exercice 4 : Horn-SAT — 3 Points

En logique propositionnelle, on appelle *clause de Horn* une clause qui contient ***au plus un*** littéral positif. Par exemple, les clauses suivantes sont des clauses de Horn : "$\neg y \vee \neg z \vee \neg x \vee \neg u \vee v$", "$w$" et "$\neg v \vee \neg x$". A l'inverse la clause suivante n'est **pas** une clause de Horn car elle contient deux littéraux positifs : "$\neg y \vee \boldsymbol{z} \vee \neg x \vee \neg u \vee \boldsymbol{v}$". Montrer que le problème suivant est dans **P**, en donnant un algorithme polynomial.

ENTRÉE : Une conjonction $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ de clauses de Horn.
QUESTION : Cette conjonction de clauses est-elle satisfaisable ?

Indication

Une clause de Horn peut être de 3 types : une variable, une disjonction de négations de variables, ou une disjonction d'une variable et de négations de variables. Dans ce dernier cas, on peut écrire la clause comme une implication ne faisant intervenir que des variables, sans négation.

Exercice 5 : Machines de Turing unidirectionnelles — 10 Points

Toutes les machines considérées dans cet exercice sont des machines de décision : elles définissent un langage. Ce sont des machines à une seule bande. De plus, sauf indication contraire, toutes les machines que nous considérons dans l'exercice (ainsi que celles qui sont demandées) peuvent être potentiellement **non-déterministes**. C'est très important : certaines questions (faciles) deviennent difficiles si on s'impose le déterminisme.

Intuitivement, une machine de Turing est *unidirectionnelle* si elle déplace toujours sa tête de lecture vers la droite et si elle s'arrête dès qu'elle lit le caractère "□" (c'est-à-dire dès qu'elle a fini de lire son entrée). Formellement, on dit qu'une machine (potentiellement non-déterministe) $M = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ est *unidirectionnelle* si son ensemble de transitions $\delta \subseteq (Q \setminus \{q_a, q_r\}) \times \Gamma \times Q \times \Gamma \times \{\triangleleft, \triangleright, \nabla\}$ satisfait les deux conditions suivantes :

— Pour toute transition $(q, a, q', b, D) \in \delta$, on a $D = \triangleright$.

— Pour toute transition (q, a, q', b, D), si $a = \square$, alors $q' = q_a$ ou $q' = q_r$.

Simplification

Puisque la machine se déplace toujours vers la droite, peu importe ce qu'elle écrit. Ainsi, on peut simplifier l'ensemble de transitions δ en le présentant comme un sous-ensemble de $(Q \setminus \{q_a, q_r\}) \times \Gamma \times Q$: peu importe ce qu'on écrit, et, également, inutile de préciser la direction puisque c'est toujours "$\triangleright$". Cette remarque vous permettra de simplifier vos réponses.

Étant donnée une machine de Turing unidirectionnelle $M = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$, on notera $L(M) \subseteq \Sigma^*$ le langage de tous les mots acceptés par M.

Partie I : Exemple et propriétés simples.

1) On considère l'alphabet $\Sigma = \{a, b\}$. Donner une machine de Turing unidirectionnelle M telle que $L(M) \subseteq \Sigma^*$ est le langage des mots de longueur paire dans Σ^*.
2) Étant données deux machines de Turing unidirectionnelles M_1 et M_2 sur le même alphabet Σ, expliquer comment construire une nouvelle machine unidirectionnelle M telle que $L(M) = L(M_1) \cup L(M_2)$. On rappelle que les machines peuvent être *non-déterministes* (la question serait plus difficile sinon).
3) Étant donnés deux langages $H, K \subseteq \Sigma^*$, on note $H \cdot K = \{uv \mid u \in H \text{ et } v \in K\}$. Autrement dit, $H \cdot K$ contient tous les mots obtenus en concaténant un mot de H avec un mot de K.

   Étant données deux machines de Turing unidirectionnelles M_1 et M_2 sur le même alphabet Σ, expliquer comment construire une nouvelle machine unidirectionnelle M telle que $L(M) = L(M_1) \cdot L(M_2)$.

Partie II : Déterminisme et complémentation.

1) Étant une machine de Turing unidirectionnelle M, expliquer comment construire une nouvelle machine unidirectionnelle M' qui est **déterministe** et telle que $L(M') = L(M)$.

Indication

La difficulté ici est que comme M peut être non-déterministe, plusieurs exécutions sont a priori possibles pour un même mot d'entrée $w \in \Sigma^*$. Ce n'est plus le cas pour la machine M' qu'on doit construire : elle ne disposera que d'une seule exécution pour un mot d'entrée donné.

La seule mémoire d'une machine unidirectionnelle est son ensemble d'états. Il faut donc doter M' d'un ensemble d'états plus grand que celui de M qui lui permet de simuler toutes les exécutions possibles de M en une seule exécution de M'. Si Q est l'ensemble d'états de M, on pourra utiliser l'ensemble des sous-ensembles de Q comme ensemble d'états de M'.

2) Étant une machine de Turing unidirectionnelle M sur l'alphabet Σ, expliquer comment construire une nouvelle machine unidirectionnelle M' telle que $L(M') = \Sigma^* \setminus L(M)$ (le langage de M' doit être le complément de celui de M').

Partie III : Problèmes de décision classiques.

1) Montrer que le problème de décision suivant est décidable :

ENTRÉE : Un alphabet Σ et une machine de Turing unidirectionnelle M sur Σ.
QUESTION : Est-ce que $L(M) \neq \emptyset$?

Ce problème est-il dans **P**? On ne demande pas de preuve formelle mais une **justification précise** aux deux réponses données.

2) Le problème de décision suivant est-il décidable?

ENTRÉE : Un alphabet Σ et une machine de Turing M (pas nécessairement unidirectionnelle) sur Σ.
QUESTION : Existe-t-il une machine de Turing unidirectionnelle M' telle que $L(M) = L(M')$?

On ne demande pas de preuve formelle mais une **justification précise**.

Partie IV : Expressions sans-étoile.

Étant donné un alphabet Σ, une *expression sans-étoile* E est un élément de syntaxe qui permet de définir un langage sur Σ^* qu'on notera $L(E)$. On définit les expressions sans-étoile par induction :

— "$\emptyset$" est une expression sans-étoile et $L(\emptyset) = \emptyset$..
— Si $a \in \Sigma$ est une lettre, alors "a" est une expression sans-étoile et $L(a) = \{a\}$.
— Si E_1, E_2, alors $E_1 + E_2$ est une expression sans-étoile et $L(E_1 + E_2) = L(E_1) \cup L(E_2)$.
— Si E_1, E_2 sont deux expressions sans-étoile, alors $E_1 \cdot E_2$ est une expression sans-étoile et $L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2)$.
— Si E est une expression sans-étoile, alors $\overline{E}$ est une expression sans-étoile et $L(\overline{E}) = \Sigma^* \setminus L(E)$.

Notation

Pour la lisibilité on omettra les parenthèses et le symbole "$\cdot$" quand il n'y a pas d'ambiguïté. Par exemple, on écrit $\overline{ab + cb}$ pour l'expression $\overline{(a \cdot b) + (c \cdot b)}$.

1) On considère l'alphabet $\Sigma = \{a, b\}$. Décrire en français les langages définis par les expressions suivantes :

$$\overline{\emptyset}, \qquad a\overline{\emptyset}, \qquad \overline{\overline{\emptyset} b \overline{b\emptyset}}$$

2) On considère l'alphabet $\Sigma = \{a, b\}$ et le langage $K = \{(ab)^n \mid n > 0\}$: K contient tous les mots non-vides de la forme $ababababab \cdots ab$. Donner une expression sans-étoile qui définit K.

3) Montrer que le problème de décision suivant est décidable. :

ENTRÉE : Un alphabet Σ et une expression sans-étoile E sur Σ.
QUESTION : Est-ce que $L(E) \neq \emptyset$?

On pourra réutiliser les résultats des trois parties précédentes (c'est même **très** fortement conseillé, elles sont là pour ça).

4) Quelle est la complexité de l'algorithme que vous avez donné pour la question précédente?

Partie V : Un peu de NP-complétude pour finir.

Étant donné un alphabet Σ et un mot $w \in \Sigma^*$, on note **alph**$(w) \subseteq \Sigma$ l'ensemble des lettres contenues dans w. Par exemple, si $\Sigma = \{a, b, c, d\}$, **alph**$(aabbaddba) = \{a, b, d\}$. On considère le problème de décision suivant :

ENTRÉE : Un alphabet Σ et deux machines de Turing unidirectionnelles M_1 et M_2 sur Σ.
QUESTION : Est-ce qu'il existe $w_1 \in L(M_1)$ et $w_2 \in L(M_2)$ tels que $\textbf{alph}(w_1) = \textbf{alph}(w_2)$?

1) Montrer que ce problème est dans **NP**.

2) Montrer que ce problème est **NP**-difficile.

Indication

Le problème à réduire est 3-**SAT**. 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 machines unidirectionnelles M_1 et M_2 sur Σ. Intuitivement, on doit construire Σ, M_1 et M_2 pour que les conditions suivantes soit satisfaites :

- Pour tout mot $w \in \Sigma^*$ accepté par M_1, $\textbf{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$.
- Pour tout mot $w \in \Sigma^*$ tel que $\textbf{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$, si w est accepté par M_2, alors l'affectation encodée satisfait toutes les clauses $C_1, \ldots, C_n$.

3) Au vu des deux premières questions, quelle est la conclusion attendue ?