| université de BORDEAUX | Année Universitaire 2020–2021<br>Master 1<br>**Théorie de la complexité**<br>**Devoir à la maison à rendre pour le 23 novembre 2020** | Collège Sciences et Technologies |
|---|---|---|

**Détection de plagiat**

Il est bien sûr autorisé de discuter entre vous de vos idées, **pas de copier un raisonnement** (même en le maquillant). En cas de plagiat, même sur une seule question d'un exercice, les étudiants concernés en assumeront les conséquences.

## Exercice 1 : 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) $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ 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 $q' = q_r$.

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

# Partie I: Exemple et propriétés simples.

1. On considère l'alphabet $\Sigma = \{a, b\}$. Donner une machine de Turing unidirectionnelle $\mathscr{M}$ telle que $L(\mathscr{M}) \subseteq \Sigma^*$ est le langage des mots de longueur paire dans $\Sigma^*$.

On considère maintenant un autre modèle de machine, appelé **machine sans mémoire**. Une telle machine $\mathscr{M}$ est donnée par un tuple $(\Sigma, Q, I, F, \delta)$ où $\Sigma$ est un alphabet fini, $Q$ un ensemble fini d'états, $I \subseteq Q$ est l'ensemble des états initiaux, $F \subseteq Q$ est l'ensemble des états finaux, et $\delta \subseteq Q \times \Sigma \times Q$ est la *relation de transition*. On note le fait que $(p, a, q) \in \delta$ par $p \xrightarrow{a} q$.

On dit qu'il existe une **exécution** de la machine de l'état $p$ à l'état $q$ sur le mot $w = a_1 \cdots a_n$ (où les $a_i$ sont des lettres de $\Sigma$) s'il existe des états $q_0, q_1, \ldots, q_n$ tels que,

$$p = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n = q.$$

On écrira alors,

$$p \xrightarrow{w} q.$$

Le langage $L(\mathscr{M})$ de la machine sans mémoire $\mathscr{M}$ est l'ensemble:

$$L(\mathscr{M}) = \left\{ w \in \Sigma^* \mid \text{il existe } q_i \in I \text{ et } q_f \in F \text{ tels que } q_i \xrightarrow{w} q_f \right\}.$$

2. Donner une machine sans mémoire dont le langage est celui des mots de longueur paire sur l'alphabet $\Sigma = \{a, b\}$.
3. Montrer que pour toute machine de Turing unidirectionnelle $\mathscr{M}$, il existe une machine sans mémoire $\mathscr{M}'$ telle que $L(\mathscr{M}) = L(\mathscr{M}')$. Inversement, montrer que pour toute machine sans mémoire $\mathscr{N}$, il existe une machine de Turing unidirectionnelle $\mathscr{N}'$ telle que $L(\mathscr{N}) = L(\mathscr{N}')$. Justifier que les constructions précédentes sont calculables.

On dit qu'un langage est *Büchi* s'il est de la forme $L(\mathscr{M})$, pour $\mathscr{M}$ une machine sans mémoire (ou pour $\mathscr{M}$ une machine unidirectionnelle, ce qui est équivalent d'après la question précédente).

4. Montrer que l'union de deux langages Büchi est un langage Büchi.
5. Montrer que l'intersection de deux langages Büchi est un langage Büchi.

6. 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 $K$.

   Montrer que si $H$ et $K$ sont des langages Büchi, alors $H \cdot K$ est aussi un langage Büchi.

## Partie II: Déterminisme et complémentation.

On dit qu'une machine sans mémoire est *déterministe* si

- elle n'a qu'un seul état initial, et
- pour tout état $p$ et pour toute lettre $a$, il y a **au plus** un état $q$ tel que $p \xrightarrow{a} q$.

1. Étant donnée une machine de Turing sans mémoire $\mathscr{M}$, expliquer comment construire une nouvelle machine sans mémoire $\mathscr{M}'$ qui est **déterministe** et telle que $L(\mathscr{M}') = L(\mathscr{M})$.

2. Étant donnée une machine de Turing sans mémoire $\mathscr{M}$ sur l'alphabet $\Sigma$, expliquer comment construire une nouvelle machine sans mémoire $\mathscr{M}'$ telle que
$$L(\mathscr{M}') = \Sigma^* \setminus L(\mathscr{M})$$
   c'est-a-dire que le langage de $\mathscr{M}'$ doit être le complément de celui de $\mathscr{M}$.

## Partie III: Problèmes de décision classiques.

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

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

   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 $\Sigma$ et une machine de Turing $\mathscr{M}$ (pas nécessairement unidirectionnelle) sur $\Sigma$.
   **QUESTION** : Existe-t-il une machine de Turing unidirectionnelle $\mathscr{M}'$ telle que $L(\mathscr{M}) = L(\mathscr{M}')$?

   On demande une **justification précise**.

## Partie IV: Expressions sans-étoile.

Étant donné un alphabet $\Sigma$, une *expression sans-étoile* $E$ est un élément de syntaxe qui permet de définir un langage sur $\Sigma^*$ 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 $\overline{ab} + c\,\overline{b}$ pour l'expression $\overline{(a \cdot b)} + (c \cdot \overline{b})$.

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

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

2. On considère l'alphabet $\Sigma = \{a, b\}$ et le langage $K = \left\{(ab)^n \mid n > 0\right\}$. Donner une expression sans-étoile qui définit $K$.

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

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

On pourra réutiliser les résultats des trois parties précédentes (c'est même **très** fortement conseillé).

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

| | | |
|---|---|---|
| **ENTRÉE** | : | Un alphabet $\Sigma$ et deux machines de Turing unidirectionnelles $\mathscr{M}_1$ et $\mathscr{M}_2$ sur $\Sigma$. |
| **QUESTION** | : | Est-ce qu'il existe $w_1 \in L(\mathscr{M}_1)$ et $w_2 \in L(\mathscr{M}_2)$ tels que $\mathrm{alph}(w_1) = \mathrm{alph}(w_2)$? |

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 $\Sigma$ et deux machines unidirectionnelles $\mathscr{M}_1$ et $\mathscr{M}_2$ sur $\Sigma$. Intuitivement, on doit construire $\Sigma, \mathscr{M}_1$ et $\mathscr{M}_2$ pour que les conditions suivantes soit satisfaites:

- Pour tout mot $w \in \Sigma^*$ accepté par $\mathscr{M}_1$, $\mathrm{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$.
- Pour tout mot $w \in \Sigma^*$ tel que $\mathrm{alph}(w)$ encode une affectation des variables $x_1, \ldots, x_k$, si $w$ est accepté par $\mathscr{M}_2$, 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?

## Exercice 2 : Logiques sur les entiers naturels (10 points)

On considère des formules exprimant des énoncés arithmétiques portant sur les entiers naturels. Un tel énoncé peut être vrai ou faux. Voici trois exemples de formules $\varphi_1$, $\varphi_2$ et $\varphi_3$:

$$\begin{aligned}
\varphi_1 &= \forall x \exists y \quad (y = x + 1). \\
\varphi_2 &= \exists x \exists y \exists z \quad (x^2 + y^2 = z^2 \wedge \neg(x\,y\,z = 0)). \\
\varphi_3 &= \exists x \exists y \exists z \quad (x^3 + y^3 = z^3 \wedge \neg(x\,y\,z = 0)).
\end{aligned}$$

Dans ces exemples, on a quantifié des variables (ici, $x, y, z$) existentiellement ($\exists$) ou universellement ($\forall$). Les variables sont interprétées comme des ***entiers naturels***. Ces exemples de formules utilisent les constantes 0 et 1, l'égalité, l'addition, la multiplication (présente dans les deux dernières formules), ainsi que des connecteurs logiques (ici, $\neg$ et $\wedge$).

La formule $\varphi_1$ énonce que tout entier naturel a un successeur: elle est vraie. La formule $\varphi_2$ est également vraie, comme le montre le choix $x = 3$, $y = 4$ et $z = 5$ (on a bien $3^2 + 4^2 = 5^2$ et $3 \times 4 \times 5 \neq 0$). Enfin, la formule $\varphi_3$ est fausse (c'est un cas très particulier du grand théorème de Fermat qu'on peut montrer directement).

Certaines variables peuvent aussi être ***libres***, c'est-à-dire non quantifiées. Par exemple, la formule,

$$\varphi_4(x) \quad = \quad \exists y \quad (x = y + y)$$

a une variable libre $x$, ce qu'on indique explicitement en la notant $\varphi_4(x)$. L'autre variable $y$ est quantifiée: on dit qu'elle est ***liée***. La valeur de $\varphi_4$ dépend de celle de $x$: si $x$ pair, la formule est vraie (car dans ce cas, il existe un entier naturel $y$ tel que $x = y + y$). Au contraire, si $x$ est impair, $\varphi_4(x)$ est fausse. La formule $\varphi_4$ est donc vraie si et seulement si $x$ est pair: elle exprime ainsi la propriété « *l'entier naturel $x$ est pair* ». Plus généralement, lorsqu'une formule $\varphi$ a des variables libres $y_1, \ldots, y_k$, on la note aussi $\varphi(y_1, \ldots, y_k)$, et sa valeur de vérité dépend des entiers naturels $y_1, \ldots, y_k$: elle définit une propriété sur les $k$-uplets d'entiers.

Cet exercice considère plusieurs syntaxes permises pour les formules. La première est la suivante: une formule est de la forme,

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \, \varphi,$$

où chaque $Q_i$ est soit $\exists$, soit $\forall$, et où $\varphi$ est une formule sans quantificateur, construite à partir de formules « *de base* » en utilisant les connecteurs logiques ($\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$). Les formules de base sont de la forme $t = t'$, où $t$ et $t'$ sont appelés des « ***termes*** ». Chaque terme est une somme de variables. Par exemple, $(x = x + x + x)$ et $(x + y = z + z)$ sont des formules de base ($x, y, z$ sont des variables). Une variable de $\varphi$ peut être,

- soit l'une des variables quantifiées $x_1, \ldots, x_n$: on dit qu'une telle variable est ***liée***.
- soit une autre variable: dans ce cas, elle n'est liée a aucun quantificateur $Q_i$, on dit qu'elle est ***libre***.

**Note**. La formule $\varphi_4$ est conforme à cette syntaxe, mais pas la formule $\varphi_1$ (qui utilise la constante 1, pour l'instant interdite) ni les formules $\varphi_2$ et $\varphi_3$ (qui utilisent la constante 0 et des produits de variables). Nous allons d'abord montrer qu'on peut en fait s'autoriser les constantes.

## Partie I: Exemples de propriétés.

1. Soit $k$ un entier naturel. Définir par récurrence sur $k$ une formule $\alpha_k(x)$, avec une variable libre $x$, telle que $\alpha_k(x)$ est vraie si et seulement si $x = k$.

La question précédente permet de généraliser la syntaxe des termes à des combinaisons affines de variables, c'est-à-dire de la forme $k + \sum_{i=1}^{p} k_i x_i$ où $k, k_i \in \mathbb{N}$ (sans pour autant augmenter l'ensemble des propriétés mathématiques exprimables). Par exemple, la formule $(x = 3y + 2)$ sécrit, avec les formules de base précédentes, comme $\exists z(\alpha_2(z) \wedge (x = y + y + y + z))$. On a utilisé une variable $z$ et $\alpha_2$ pour définir la constante 2. On permet donc les combinaisons affines de variables à partir de maintenant.

2. Écrire une formule à deux variables libres $x, y$ qui exprime la propriété $x \leqslant y$. Vous pouvez utiliser une ou plusieurs autres variables (liées). De même, écrire une formule qui exprime la propriété $x < y$.

La question 2 permet désormais d'utiliser les comparaisons de termes $t < t'$, $t \leqslant t'$, $t > t'$ et $t \geqslant t'$.

3. Exprimez en français ce qu'énonce la formule $\forall x \exists y \exists z \; (y > x) \wedge (y = 2z + 1)$. Est-elle vraie?

> **Attention!**
>
> On ne demande pas une paraphrase de cette formule (du type « pour tout entier $x$, il existe un entier $y$ et un entier $z$ tels que... ») mais d'expliquer quelle propriété des entiers naturels est exprimée.

Dans les deux questions suivantes, on autorise les formules de base de la forme $x = yz$, où $x, y, z$ sont des variables, en plus des formules de base précédentes.

4. Quelle est la propriété sur $z$ exprimée par la formule $\kappa(z) = \exists x \exists y \; (x > 1) \wedge (y > 1) \wedge (z = xy)$?

5. Écrire une formule exprimant qu'il existe une infinité de couples $(p, p+2)$ où $p$ et $p+2$ sont premiers.

Dans le reste de l'exercice, on s'intéresse au problème de calculer la valeur de vérité de telles formules sans variable libre (une telle formule est donc vraie ou fausse), du point de vue décidabilité et complexité. Dans les parties II et III,

on considère ce problème en restreignant, de façon différente dans chaque partie, l'ensemble des formules d'entrée par la syntaxe autorisée.

## Partie II: Logique avec égalité seulement.

On considère maintenant des formules dont la syntaxe est restreinte: les seuls termes autorisés sont les constantes (codées en binaire) et les variables. Ainsi, $x = y$, $x = 0$, $x = 1$ et $x = 42$ sont des formules de base.

1. Quelle est la complexité la plus précise que vous connaissez contenant le problème suivant?

   **Entrée** : Une formule $\varphi$ dont la syntaxe est restreinte comme ci-dessus, sans variable libre.
   **Question**: La formule $\varphi$ est-elle vraie?

2. Même question si on suppose en plus que chaque quantificateur de la formule d'entrée est $\exists$.

On revient maintenant à la « *logique avec addition* », dans laquelle les termes autorisés sont des combinaisons affines de variables.

## Partie III: Logique avec addition et multiplication.

Dans cette partie, on étend la syntaxe de la façon suivante: un terme est un ***produit*** d'une ou plusieurs combinaisons affines de variables à coefficients entiers naturels. Par exemple, $5z + 1 = (2x + 1)(3y + 2)$ est une formule de base autorisée. On appelle cette logique « *logique avec addition et multiplication* ». Les formules $\varphi_1$, $\varphi_2$, $\varphi_3$ et $\varphi_4$ sont conformes à cette syntaxe. L'objectif est de montrer que le problème « **Logique Add-Mult** » suivant est indécidable:

**Entrée** : Une formule $\varphi$ sans variable libre dans la logique avec addition et multiplication.
**Question**: La formule $\varphi$ est-elle vraie?

À tout mot $w$ sur l'alphabet $\{1,2\}$, on associe sa valeur $\overline{w}$ en base dix. Par convention, la valeur du mot vide est $\overline{\varepsilon} = 0$. Ainsi par exemple, $\overline{121}$ vaut l'entier 121 (cent vingt et un).

1. Que peut-on dire de $u$ et $v$ lorsque $\overline{u} = \overline{v}$?
2. Soit $u, w$ deux mots de $\{1,2\}^*$. On pose $w = a_1 \cdots a_n$. Exprimer $\overline{uw}$ en fonction de $\overline{u}$, $\overline{w}$ et de $n$.
3. Soit $w \in \{1,2\}^*$. Écrire une formule $f_w(x, y)$ à deux variables libres $x, y$ qui exprime que l'entier $y$ est la valeur du mot obtenu en concaténant $w$ à la suite de la représentation décimale de $x$. Par exemple, si $w$ est le mot 2222, $f_w(x, y)$ est vraie pour les entiers $x = 121$ et $y = 1212222$ (et, pour cette valeur de $x$, $f_w(x, y)$ n'est vraie pour aucune autre valeur de $y$).
4. On suppose dans cette question seulement que l'on peut quantifier l'existence d'une suite finie d'entiers naturels. Autrement dit, on peut écrire $\exists n \, \exists (m_k)_{k \leqslant n} \varphi$. Montrer que le problème suivant est indécidable:

   **Entrée** : Une formule $\varphi$ sans variable libre dans la logique avec addition, multiplication, et quantification sur les suites.
   **Question**: La formule $\varphi$ est-elle vraie?

**Indication**

Utiliser les questions 1 et 3 pour réduire le problème de correspondance de Post à ce problème.

On veut maintenant se passer de la quantification sur les suites finies pour montrer l'indécidabilité du problème **Logique Add-Mult**. On utilise le théorème des restes chinois dont l'énoncé est le suivant.

> Soient $n_1, \ldots, n_k$, $x_1, \ldots, x_k$ des entiers naturels. On suppose que si $i \neq j$, $n_i$ et $n_j$ sont premiers entre eux. Alors, il existe un entier $x$ tel que $x \equiv x_i \pmod{n_i}$ pour tout $1 \leqslant i \leqslant k$.

Soit $(x_1, \ldots, x_k)$ une suite finie d'entiers naturels et $n = (\max(k, x_1, \ldots, x_k))!$, où $p!$ désigne la factorielle de $p$. Pour chaque $i = 1, \ldots, k$, soit $n_i = in + 1$.

5. Montrer que si $i \neq j$, les entiers $n_i$ et $n_j$ sont premiers entre eux.
6. Soit $x$ un entier donné par le théorème des restes chinois. Montrer que chaque $x_i$ peut s'exprimer en fonction de $x, n, i$ en utilisant uniquement l'addition et la multiplication.
7. Déduire des questions précédentes que le problème **Logique Add-Mult** est indécidable.