université de BORDEAUX	**Année Universitaire 2021–2022, Master CSI** Théorie de la complexité (UE 4TCY701U) À rendre sous Moodle d'octobre à décembre 2021	**Collège Sciences et Technologies**

Problèmes de décision sur les langages

Présentation du sujet

Le sujet est un problème qui comporte plusieurs parties indépendantes.

- La première partie traite de propriétés algorithmiques de grammaires.
- La seconde établit l'indécidabilité d'un problème sur les mots, afin d'en déduire des conséquences sur les langages.
- Le sujet sera complété par une troisième partie spécifique à la complexité.

Remise et dates limites

Les parties du problème sont indépendantes. Elles sont à déposer sous Moodle :

- Dépôt de la première partie avant le 25/10/2021.
- Dépôt de la seconde partie avant le 8/11/2021.

Conseils pour la rédaction

- Rédigez soigneusement chaque question, dans un français correct.
- Justifiez vos affirmations : une réponse non ou mal justifiée n'apporte pas de point.
- Vérifiez vos algorithmes et affirmations sur des cas particuliers (*safety checks*).
- Les questions d'une même partie sont liées entre elles : il arrive souvent qu'une question serve dans la résolution d'une ou plusieurs autres questions. Vous pouvez utiliser toute question, même non faite, pour répondre à une autre question.
- Enfin, pour montrer qu'un problème est *décidable*, n'explicitez **pas** une machine de Turing décidant le problème. À la place, décrivez en français un algorithme. La description des algorithmes doit être précise, mais sans entrer dans des détails d'implémentation inutiles (comme d'expliciter des structures de données). En revanche, il est important de justifier pourquoi un algorithme est correct, et en particulier pourquoi il s'arrête sur tout entrée.

Plagiat interdit

Vous pouvez partager vos idées, mais la rédaction doit être **personnelle**. En cas de plagiat avéré, même sur une seule question, les copies concernées seront notées 0.

1 Algorithmes sur les grammaires

Dans le devoir, le terme **grammaire** signifie **grammaire hors-contexte** (il existe en effet d'autres grammaires).

Rappels et notations

Soit $G = (\Sigma, V, S, R)$ une grammaire, où Σ est l'alphabet des symboles *terminaux*, V est l'ensemble des *variables* (ou *non-terminaux*), $S \in V$ est le *symbole de départ* et $R \subseteq V \times (\Sigma \cup V)^*$ est l'ensemble des *règles*. On note habituellement une règle $X \to \alpha$ plutôt que (X, α). On rappelle que :

- On étend ainsi la notation $\to$ aux mots : si β et γ sont deux mots de $(\Sigma \cup V)^*$, on note $\beta \to \gamma$ si on peut factoriser $\beta = \beta' X \beta''$ et $\gamma = \beta' \alpha \beta''$, où $X \to \alpha$ est une règle. Autrement dit, $\beta \to \gamma$ si γ peut être obtenu de β par remplacement d'une variable par le membre droit d'une règle dont cette variable est membre gauche. On dit alors que β se dérive en γ, en un pas de dérivation.
- Pour deux mots β et γ de $(\Sigma \cup V)^*$, on note $\beta \xrightarrow{*} \gamma$ si β se dérive en γ en un nombre fini de pas (ce nombre peut être nul, auquel cas $\beta = \gamma$).
- Le langage engendré par G est par définition : $\mathcal{L}(G) = \{ w \in \Sigma^* \mid S \xrightarrow{*} w \}$.

Soit L un langage hors-contexte engendré par une grammaire $G = (\Sigma, V, S, R)$. On dit qu'une variable X est **utile** s'il existe une dérivation de la forme $S \xrightarrow{*} \alpha X \beta \xrightarrow{*} w$, où α et β sont dans $(\Sigma \cup V)^*$ et $w \in \Sigma^*$. Autrement dit, on peut dériver un mot contenant X depuis le symbole de départ S, puis dériver un mot de Σ^* à partir de ce mot. Inversement, une variable qui n'est pas utile ne peut intervenir dans aucune dérivation depuis S produisant un mot de Σ^*. L'objectif des premières questions est de montrer qu'on peut déterminer les variables inutiles d'une grammaire, puis qu'on peut les supprimer sans changer le langage engendré.

1. Décrire en français un algorithme calculant l'ensemble des variables X telles qu'il existe $w \in \Sigma^*$ tel que $X \xrightarrow{*} w$.

Indication

Calculer cet ensemble en l'initialisant correctement, et en y ajoutant itérativement des variables.

2. Démontrer que l'algorithme décrit en question 1 est correct, c'est-à-dire que :
   (a) Pour toute variable X de l'ensemble calculé par votre algorithme, il existe un mot $w \in \Sigma^*$ tel que $X \xrightarrow{*} w$.
   (b) Inversement, si $X \xrightarrow{*} w$ avec $w \in \Sigma^*$, alors X est bien dans l'ensemble calculé par votre algorithme.

Indication

Raisonner par récurrence sur la longueur de la dérivation $X \xrightarrow{*} w$.

3. L'algorithme décrit en question 1 est-il polynomial ? Justifier brièvement.
4. Décrire de même un algorithme calculant l'ensemble des variables X telles qu'il existe $\alpha, \beta \in (\Sigma \cup V)^*$ tels que $S \xrightarrow{*} \alpha X \beta$. On ne demande pas de démontrer la correction de cet algorithme.
5. Déduire des questions précédentes si le problème suivant est décidable. Si oui, décrire un algorithme et dire si cet algorithme est polynomial. Sinon, faire une preuve d'indécidabilité et préciser s'il est semi-décidable ou non :

   ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$.

   QUESTION : La grammaire G engendre-t-elle le langage vide ?

6. Décrire un algorithme prenant en entrée une grammaire G, une variable X de G, et testant si $X \xrightarrow{*} \varepsilon$ (on ne demande pas de prouver qu'il est correct). Indiquer si l'algorithme est polynomial, en justifiant brièvement.
7. Déduire de la question 6 si le problème suivant est ou non décidable. Si oui, décrire un algorithme et dire si cet algorithme est polynomial. Sinon, faire une preuve d'indécidabilité et préciser s'il est semi-décidable ou non :

   ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$.

   QUESTION : Le mot vide appartient-il au langage engendré par la grammaire G ?
8. Décrire en français un algorithme prenant en entrée une grammaire G qui n'engendre que des mots non-vides, et qui calcule une grammaire G' engendrant le même langage que G, et qui n'a aucune règle de la forme $X \to \varepsilon$ (pour cette question, on ne demande pas de montrer que l'algorithme est correct).

Indication

Testez votre algorithme sur la grammaire dont les règles sont : $S \to A_1 \cdots A_n$, et pour tout $i \leqslant n$, $A_i \to a_i \mid \varepsilon$. Notez que cette grammaire engendre le même langage que la grammaire dans laquelle on remplace la première règle par les $n-1$ règles suivantes : $S \to A_1 B_1$, $B_1 \to A_2 B_2$, …, $B_{n-2} \to A_{n-1} A_n$. Pour évaluer et améliorer la complexité de votre algorithme, vous pouvez vous inspirer de cette transformation.

9. Sous les mêmes hypothèses, montrer qu'on peut construire la grammaire G' pour qu'elle n'ait ni règle de la forme $X \to \varepsilon$, ni règle de la forme $X \to Y$, où X et Y sont des variables.
10. Soit G une grammaire qui n'a ni règle de la forme $X \to \varepsilon$, ni règle de la forme $X \to Y$, où X et Y sont des variables. Soit $w \in \Sigma^*$ un mot tel que $S \xrightarrow{*} w$. Calculer une borne sur la longueur de cette dérivation en fonction de la longueur du mot w, et justifier que cette borne est correcte.
11. Déduire de la question 10 si le problème suivant est ou non décidable. Si oui, décrire un algorithme et dire si cet algorithme est polynomial. Sinon, faire une preuve d'indécidabilité et préciser s'il est semi-décidable ou non :

    ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$ et un mot w.

    QUESTION : La grammaire G engendre-t-elle le mot w ?
12. Soit G une grammaire sans variable inutile et sans règle de la forme $X \to \varepsilon$. Montrer que G engendre un langage infini si et seulement si il existe une variable X et des mots $\alpha, \beta \in (\Sigma \cup V)^*$ tels que α ou β est non vide, et $X \xrightarrow{*} \alpha X \beta$.
13. Déduire de la question 12 si le problème suivant est ou non décidable. Si oui, décrire un algorithme et dire si cet algorithme est polynomial. Sinon, faire une preuve d'indécidabilité et préciser s'il est semi-décidable ou non :

    ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$ et un mot w.

    QUESTION : La grammaire G engendre-t-elle un langage infini ?
14. Parmi les problèmes suivants, deux exactement sont semi-décidables. Lesquels (justifiez) ?

    (a) **Intersection de langages hors-contextes**

    ENTRÉE : Deux grammaires G_1 et G_2 sur le même alphabet Σ.

    QUESTION : Y a-t-il un mot dans l'intersection des langages $\mathcal{L}(G_1)$ et $\mathcal{L}(G_2)$?

    (b) **Universalité pour langage hors-contexte**

    ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$.

    QUESTION : Le langage engendré par G est-il Σ^* ?

    (c) **Non-universalité pour langage hors-contexte**

    ENTRÉE : Une grammaire $G = (\Sigma, V, S, R)$.

    QUESTION : Le langage engendré par G est-il différent de Σ^* ?

2 Le problème de tuilage par mots

Définition 1 (Tuiles)

- Dans cette partie, on appelle **tuile** (sur un alphabet Σ) un couple de mots $(v, w) \in \Sigma^* \times \Sigma^*$.

Définition 2 (Problème de tuilage par mots)

Le problème de tuilage par mots (**PTM** en abrégé) est le suivant. Une entrée de ce problème est composée d'un entier $k \geqslant 1$ et d'une suite de k tuiles sur Σ : $(v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k)$. Une solution au problème est une suite finie d'indices $i_1, i_2, \ldots, i_n$, avec $n \geqslant 1$ et $1 \leqslant i_j \leqslant k$ pour tout j, telle que,

$$v_{i_1} v_{i_2} \cdots v_{i_n} = w_{i_1} w_{i_2} \cdots w_{i_n}.$$

Le problème consiste à déterminer si une solution existe ou non.

1. Pour chacune des entrées suivantes, dire si le PTM a une solution. Si oui, donner une solution (pour ces entrées, en chercher de longueur 10 au maximum). Sinon, expliquer pourquoi il n'y a pas de solution.

(a)

i	1	2	3
v_i	b	aab	ba
w_i	a	$abba$	a

(b)

i	1	2	3
v_i	ab	aab	a
w_i	a	ab	a

(c)

i	1	2	3
v_i	aa	a	ba
w_i	ab	aaa	a

(d)

i	1	2	3
v_i	a	aa	ba
w_i	aaa	ab	a

(e)

i	1	2	3	4
v_i	aab	aa	b	bab
w_i	aa	ab	a	ba

On veut maintenant prouver que le PTM est indécidable. Pour cela, on définit la variante suivante de ce problème.

Définition 3 (Problème de tuilage par mots contraint)

Le problème de tuilage par mots contraint (**PTMC** en abrégé) est le suivant. Une entrée de ce problème est identique à l'une du PTM. Une solution au problème est une suite finie d'indices $i_1, i_2, \ldots, i_n$, avec $n \geqslant 1$ et $1 \leqslant i_j \leqslant k$ pour tout j, telle que,

$$v_{i_1} v_{i_2} \cdots v_{i_n} = w_{i_1} w_{i_2} \cdots w_{i_n}$$

avec $i_1 = 1$. Le problème consiste à déterminer si une solution existe ou non.

2. Quelles sont instances de la question 1 qui sont solutions de PTMC?
3. Expliquer pourquoi la fonction identité n'est pas une réduction du problème PTMC au problème PTM.
4. Montrer que le problème PTMC se réduit au problème PTM.

Indication

Commencer par revenir à la définition d'une réduction : écrire ce que cela signifie, et écrire l'objectif, en précisant ce qui est donné, ce qu'il faut construire et ce qu'il faut assurer comme propriété. Il peut être utile d'introduire une nouvelle lettre $\diamond$, et, pour $u \in \Sigma^*$, de considérer les mots suivants de $(\Sigma \cup \{\diamond\})^*$:

- Le mot $u^\diamond$, obtenu à partir de u en insérant un $\diamond$ entre chaque paire de lettres consécutives de u (par exemple, $(a\,b\,c)^\diamond = a \diamond b \diamond c$).
- Les mots $\diamond u^\diamond$, $u^\diamond \diamond$ et $\diamond u^\diamond \diamond$.

5. Que peut-on déduire de la question 4 si on prouve que PTM est indécidable ? Justifier brièvement.
6. Que peut-on déduire de la question 4 si on prouve que PTMC est indécidable ? Justifier brièvement.

On veut maintenant réduire le problème de l'arrêt sur mot vide d'une machine de Turing au problème PTMC.

7. Écrire ce qu'il faut faire pour cela.

Pour faire la réduction, on commence par représenter chaque configuration de machine de Turing par un mot, de la façon suivante. Quitte à renommer les états de la machine, on peut supposer que l'ensemble des états est disjoint de l'alphabet de travail. Soit C une configuration. On note :

- u le mot écrit sur la bande jusqu'à la tête de lecture (exclue).
- v le mot écrit sur la bande à partir de la tête de lecture (inclue) et jusqu'au premier $\square$ à droite de cette tête.

Si la machine est dans l'état q, le mot uqv décrit complètement la configuration C (ce mot sera noté C également). En effet, la seule lettre de ce mot qui est dans l'ensemble des états est q. Par exemple, considérons la configuration suivante : le mot écrit sur la bande est $\$abadaa$, la machine est dans l'état q_7 et sa tête est sur l'unique lettre d. Cette configuration sera représentée par le mot $\$abaq_7daa$.

L'idée de la réduction est de forcer les solutions minimales éventuelles du PTMC sur l'entrée construite par réduction à produire les concaténations de la forme :

$$(\#C_0\#C_1\#C_2\#...\#C_n, \quad \#C_0\#C_1\#C_2\#...\#C_{n+1}),$$

où les C_k sont les configurations successives de la machine de départ, et le symbole # est une nouvelle lettre (qui n'appartient pas à l'alphabet de travail de la machine de Turing). Autrement dit, on fait en sorte que le mot produit par concaténation des v_i code une exécution partielle de la machine, qui soit « en retard » d'une configuration sur le mot produit par concaténation correspondante des w_i. Des tuiles supplémentaires permettront de « rattraper ce retard » si (et seulement si) C_{n+1} contient l'état d'acceptation de la machine.

8. Une première tentative serait de construire une instance du PTMC incluant toutes les tuiles de la forme $(\#C_i, \#C_{i+1})$. Expliquer pourquoi ce n'est pas possible.
9. Que peut-on dire d'une instance du PTM qui contient une tuile de la forme (x, x) ? Peut-on conclure de même pour le PTMC ?
10. Expliquer la réduction par étapes :
    (a) Proposer une première tuile (v_1, w_1) pour l'instance à construire.
    (b) Soient C et C' deux mots codant des configurations successives de la machine, c'est-à-dire telles que $C \to C'$. En combien de positions les mots C et C' peuvent-ils différer ?
    (c) (♠) Compléter et expliquer la réduction en décrivant en français :
    - les tuiles à ajouter pour que la concaténation des w_i reste en avance d'une configuration sur la concaténation correspondante des mots v_i,
    - celles à ajouter pour permettre à la concaténation des v_i de rattraper celle des w_i si (et seulement si) l'état d'acceptation est rencontré.

11. Démontrer que le PTM restreint à l'alphabet binaire $\{0,1\}$ reste indécidable.
12. Démontrer que le PTM restreint à l'alphabet unaire $\{1\}$ est décidable.

On utilise maintenant le PTM pour déduire l'indécidabilité de certains problèmes sur les grammaires.

13. Montrer que le problème **Intersection de langages hors-contexte** de la question 14 (a) de la partie 1 est indécidable.

Indication

On peut réduire le complémentaire du PTM au problème d'intersection vide pour les langages hors-contexte. Pour cela, étant donnée une instance du PTM $(v_1, w_1), \ldots, (v_k, w_k)$ sur l'alphabet Σ, expliquer comment construire deux langages hors-contexte L_1, L_2 tels que $L_1 \cap L_2 = \emptyset$ si et seulement si l'instance du PTM n'a pas de solution.

Vous pouvez construire un nouvel alphabet $\Delta = \Sigma \cup \{1, \ldots, k\} \cup \{\#\}$ (où les lettres $1, \ldots, k$ et $\#$ sont de nouvelles lettres, *i.e.*, n'appartiennent pas à Σ) et considérer le langage suivant Δ :

$$L_v = \left\{ i_n i_{n-1} \cdots i_1 \# v_{i_1} \cdots v_{i_{n-1}} v_{i_n} \mid n \geqslant 1 \right\}.$$

14. Montrer que le problème **Universalité pour langage hors-contexte** de la question 14 (b) de la partie 1 est indécidable.

Indication

On peut à nouveau réduire le complémentaire du PTM au problème d'universalité pour langage hors-contexte. Vous pouvez considérer l'alphabet Δ de l'indication de la question précédente, et montrer que le **complémentaire** du langage L_v de cette indication est hors-contexte.

15. Le problème **Non-universalité pour langage hors-contexte** de la question 14 (b) de la partie 1 est indécidable est-il semi-décidable ? Justifier.