TP7 - IFT2105

par Ilan Elbaz

3 juillet 2019

Rappel de définitions

- **Définition 1**. On dit qu'un langage L est décidable s'il existe une Machine de Turing M telle que:
  - si $w \in L$, M accepte w
  - si $w \notin L$, M rejette w

  On dira aussi que la machine M décide le langage L
  La classe des langages décidables est fermée sous la complémentation.

- **Définition 2**. On dit qu'un langage L est reconnaissable s'il existe une Machine de Turing M telle que:
  - si $w \in L$, M accepte w
  - si $w \notin L$, M rejette w ou boucle sur w

  On dira aussi que la machine M décide le langage L

- **Définition 3**. Un langage L tel que L et $\bar{L}$ sont reconnaissables est décidable.

  $$L_U = \{\langle M, w\rangle \mid M \text{ est une } MT \text{ et } M \text{ accepte le mot } w\}$$

  On a vu que L_U est indécidable, mais qu'il est reconnaissable. On peut conclure que $\overline{L_U}$ est non-reconnaissable.

- **Définition 4**. On dit que le langage L_1 se réduit à L_2, noté $L_1 \leq L_2$, s'il existe une fonction calculable

  $$f : \Sigma^* \to \Sigma^*$$

  tel que

  $$x \in L_1 \Leftrightarrow f(x) \in L_2$$

Le Problème de Correspondance de Post

Soit Σ un alphabet fini, et $P \subseteq \Sigma^* \times \Sigma^*$ un ensemble fini de dominos étiquetés par des mots sur l'alphabet Σ (des paires de mots, donc). Le Problème de Correspondance de Post (PCP), introduit par Emile Post en 1946, consiste à déterminer s'il existe une séquence de dominos de P tels que le mot obtenu par la concaténation des premières composantes est identique à celui formé par la concaténation des secondes composantes. Plus formellement, on cherche à déterminer l'existence d'une suite $(u_i, v_i)_{0 \leq i \leq n}$ telle que : $u_0 \cdot u_1 \cdots u_n = v_0 \cdot v_1 \cdots v_n$.

Montrez que le probleme PCP est décidable sur l'alphabet $\Sigma = \{1\}$

Le problème PCP sur l'alphabet unaire est décidable. Pour montrer cela on decrit M la MT qui decide de unaire PCP avec les instances suivantes:

$$\left\{ \left[\frac{1^{a_1}}{1^{b_1}} \right], \cdots, \left[\frac{1^{a_n}}{1^{b_n}} \right] \right\}$$

M prend en entrée $\langle a_1, b_1, \cdots, a_n, b_n \rangle$:

- Verifier si il existe des dominos tel que $a_i = b_i$ si oui accepter.
- Verifier si il existe i, j tel que $a_i > b_i$ et $a_j < b_j$ si oui accepter, sinon rejeter.

Dans la première étape, M verifie si il existe un domino qui a lui seul est une solution au problème.
Dans la seconde étape, M regarde si il une paire de domino qui forme une solution. Si il arrive à trouver une telle paire, on peut construire une solution en prenant $(b_j - a_j)$ pièces du i^e domino, et en prenant $(a_i - b_i)$ du j^e domino. Cette construction aura ainsi $a_i(b_j - a_j) + a_j(a_i - b_i) = a_i b_j - a_j b_i$ "1" en haut et $b_i(b_j - a_j) + b_j(a_i - b_i) = a_i b_j - a_j b_i$ "1" en bas.
Si tout les dominos on leurs parties du haut avec plus de uns que leurs partie du bas dans ce cas il n'y aura pas de solution et la machine rejettera.

Reconnaissabilité de L et $\bar{L}$

Un état q d'une machine de Turing est utile s'il existe un mot d'entrée $w \in \Sigma^*$ sur lequel le calcul de la machine passe par l'état q . Considérez le langage:

$$L = \{\langle M \rangle : M \text{ possède un état inutile}\}$$

L n'est pas reconnaissable. On va montrer que $\overline{L_U} \leq L$
Définissons une machine S:

S prend en entrée y,

1. Si y n'est pas de la forme $\langle M, w \rangle$, bâtire M' où M' a un état inutile (Par exemple, une machine rejettant tous les mots qui n'utilise pas son état acceptant).
2. Sinon, construire une machine M' comme copie de M mais avec les modifications suivantes à sa fonction de transition de sorte à ce que
   (a) M' fait un tour de tous ses états (sauf q_{accept} et q_{reject}).
   (b) M' compare son entrée x avec w, rejeter si $x = w$ et sinon simmuler M sur w .

Soit f la fonction calculée par S. $f(y) = M'$

Si y n'est pas de la forme $\langle M, w \rangle$, alors M' est une machine avec un état inutile.

Si $\langle M, w \rangle \in L_U$ ($\Leftrightarrow \langle M, w \rangle \notin \overline{L_U}$), alors M' est une machine de Turing qui accepte toutes les entrées sauf w qu'elle rejette. Ainsi M' est construite de sorte a se servir de tous ses états. Donc $\langle M' \rangle \notin L$

Si $\langle M, w \rangle \notin L_U$ ($\Leftrightarrow \langle M, w \rangle \in \overline{L_U}$), alors M' est une machine de Turing qui n'accepte aucune entrée. Par conséquent son état q_{accept} n'est jamais accédé et donc inutile. Donc $\langle M' \rangle \in L$

$\overline{L}$ est il reconnaissable?
Oui $\overline{L}$ est il reconnaissable. Considérons la machine de Turing M suivante:

1. Si y n'est pas de la forme $\langle M \rangle$ accepter.
2. Soit $m_1, m_2, ...$ les mots Σ^* dans l'ordre lexicographique.
3. Pour $j = 1, 2, ...$: exécuter M pour j étapes sur les mots $m_1, m_2...$ et marquer les états rejoints; si tous les états ont été marqués, accepter"

Si y n'est pas l'encodage d'une MT, alors $y \in L$ et S accepte.

Si M n'a pas d'états inutiles, alors il existe un j minimal tel que M passe par tous ses états en j étapes pour les j premiers mots. S reconnait L.

Montrez que $FINI_{MT}$ est indécidable

$$FINI_{MT} = \{\langle M \rangle | M \text{ est une MT et } L(M) \in FINI\}$$

On montre qu'il est indécidable en montrant que $L_U \leq \overline{FINI_{MT}}$.
Démonstration. Donnons une fonction $f : \Sigma^* \to \Sigma^*$ calculable telle que $y \in L_U \Leftrightarrow f(y) \in \overline{FINI_{MT}}$
Définissons f comme la fonction calculée par la machine S suivante :
S prend en entré y

1. Si y n'est pas de la forme $\langle M, w \rangle$ bâtir une machine M' qui rejette tout les mots (même le mot vide).
2. Sinon, bâtir M':
   (a) prendre le mot x en entrée
   (b) Simuler M sur w
   (c) Si M accepte, alors
      i. si $x = a^n$ alors accepter
      ii. sinon rejeter
   (d) si M rejette rejeter

Si $y \in L_U$, alors $y = \langle M, w \rangle$ et M accepte w. M' a comme langage $\{a^n\} \in \overline{FINI}$ et appartient à $\overline{FINI_{MT}}$. Si $y \notin L_U$, alors M' a comme langage $\emptyset$ et appartient à FINI .

Montrez que T est indécidable

$$T = \{\langle M \rangle \mid M \text{ est une MT qui accepte } w^R \text{ quand elle accepte } w\}$$

On montre plutôt que $\overline{T}$ est indécidable en montrant $L_U \leq \overline{T}$. Si c'est le cas alors T est aussi indécidable, car s'il l'était, on pourrait décider de son complément.

Soit une machine S
S prend en entré y:

1. Si y n'est pas de la forme $\langle M, w \rangle$ bâtir une machine M' de langage vide.
2. Sinon, construire une machine M' qui rejette tous les mots sauf $0w1$ et a le même comportement que M sur w sur entrée $0w1$.

Soit f la fonction calculée par S. Montrons $y \in L_U \Leftrightarrow f(y) \in \overline{T}$.

Si y n'est pas de la forme $\langle M, w \rangle$, alors $y \notin L_U$ et M' n'accepte aucun mot. Triviallement, $M' \in T$ ($\Leftrightarrow M' \notin \overline{T}$)

Si $\langle M, w \rangle \notin L_U$, alors M n'accepte pas w et $L(M') = \emptyset$. Dans ce cas, $M' \in T$ ($\Leftrightarrow M' \notin \overline{T}$).

Si $\langle M, w \rangle \in L_U$, alors M accepte w et $L(M') = \{0w1\}$. Dans ce cas, $M' \in \overline{T}$.

Rappel de définitions

- **Définition 5**. $I \subseteq \Sigma^*$ est un ensemble d'indices si, pour toute paire $\langle M1 \rangle$ et $\langle M2 \rangle$ telle que M1 et M2 sont des MT équivalentes, c'est à dire $L(M1) = L(M2)$, on a :

  $$\langle M1 \rangle \in I \Leftrightarrow \langle M2 \rangle \in I.$$

  Autrement dit, le fait que $\langle M \rangle$ soit dans I ou non dépend uniquement du langage de M.

- **Définition 6**. (th. de Rice) Soit I un ensemble d'indices non trivial, c'est à dire :

  - il existe une MT M^* telle que $\langle M^* \rangle \in I$
  - il existe une MT $M^\dagger$ telle que $\langle M^\dagger \rangle \notin I$

  alors I est indécidable.

avec RICE

T est bien un langage constitué de descriptions de machines de Turing et il satisfait les conditions suivantes :

1. Premièrement, T est non-trivial
   (a) Il existe une machine de Turing M, telle que $\langle M \rangle \in T$, soit la machine M qui accepte les palindrômes,
   (b) Il existe une machine de Turing N telle que $\langle N \rangle \notin T$, soit la machine N qui accepte un non-palindrôme seulement.
2. Deuxièmement, T est une propriété des langages des MT. Alors le problème de déterminer si $\langle M \rangle \in T$ est indécidable.

Montrez avec Rice que T est indécidable

$$T = \{\langle M \rangle | M \text{ accepte tout les mot } w \in \{0,1\}^* \text{ de longueur pair}\}$$

1. Premièrement, T est non-trivial
   (a) Il existe une machine de Turing M, telle que $\langle M \rangle \in T$, soit la machine M qui accepte les entrée à ruban contenant un nombre pair de caractères,
   (b) Il existe une machine de Turing N telle que $\langle N \rangle \notin T$, soit la machine N qui accepte les entrée à ruban contenant un nombre impaire de caractères,
2. Deuxièmement, T est une propriété des langages des MT. Alors le problème de déterminer si $\langle M \rangle \in T$ est indécidable.

Montrez que L est indécidable

Considérez le problème de déterminer si une machine de Turing M sur entrée w essaie, à un moment ou à un autre de son exécution, de déplacer sa tête de lecture à gauche alors qu'elle est située sur la case la plus à gauche du ruban. Formulez ce problème en forme de langage L et montrez qu'il est indécidable.

L = $\{\langle M, w\rangle$: M sur entrée w essaie de déplacer sa tête à gauche lorsque celle-ci ce trouve dans la première case du ruban.$\}$

Attention, on ne peut pas utiliser le théorème de Rice, car "M sur entrée w essaie de déplacer sa tête à gauche lorsque celle-ci ce trouve dans la première case du ruban sur entrée w" n'est pas une propriété du langage $L(M)$ mais de l'exécution de M.

On va bâtir une fonction calculable f telle que $y \in L_u \Leftrightarrow f(y) \in L$
S= "Sur entrée y,

1. Si y n'est pas de la forme $\langle M, w\rangle$, effacer le ruban.
2. Sinon, construire une machine M' qui a le même comportement que M sauf sous deux conditions.
   (a) Lorsque M va dans un état acceptant, M' va plutôt déplacer la tête dans la première case et essayer de se déplacer à gauche.
   (b) Déplacer le contenu du ruban de M d'une case vers la droite. Mettre un caractère spécial dans la première case. Lorsque la tête lit ce caractère spécial dans la première case. Lorsque la tête lit ce caractère spécial (la machine a essayé de se déplacer trop à gauche), se déplacer à droite et rester dans le même état, sauf si on est dans le cas précédent."

   Soit f la fonction calculée par S.

   Si $\langle w, M\rangle \in L_U$ alors $f(\langle w, M\rangle) = M'$ entre dans un état acceptant, donc M' va aller dans la première case et essayer de se déplacer à gauche et $M' \in L$

   Si y n'est pas de la forme $\langle w, M\rangle$ alors il n'appartient pas à L_U. La machine ϵ n'est pas dans L.

   Si $\langle w, M\rangle \notin L_U$ alors M n'entre jamais dans un état acceptant. M' capture toujours les déplacement à gauche lorsque la tête se trouve dans la première case et $M' \notin L$

Problèmes dans NP ou dans P? justifiez

1.
   - Données : Un graphe $G = (V, E)$
   - Question : Existe-t-il un cycle de longueur égale à $\left\lfloor \frac{|V|}{2} \right\rfloor$?

   Le problème 1 est dans NP car étant donnée une suite de sommets s, on peut vérifier en temps polynomial si s est un cycle de de longueur égale à $\left\lfloor \frac{|V|}{2} \right\rfloor$ dans G (en $O(n^2)$ opérations).

2.
   - Données : Un graphe $G = (V, E)$
   - Question : Existe-t-il un cycle de longueur égale à 4?

   Le problème 2 est dans NP car étant donnée une suite de sommets s, on peut vérifier en temps polynomial si s est un cycle de de longueur égale à 4 dans G (en $O(n)$ opérations).
   Le problème 2 est dans P.Considérons l'algorithme suivant. Pour tout 4-uplets de sommets (x, y, z, t) vérifier s'il est un cycle de de longueur égale à 4. Si c'est un cas, pour un seul 4uplets, alors répondre oui sinon répondre non. La complexité de cet algorithme est $O(n^5)$ (il y a $O(n^4)$ 4-uplets et tester s'il est un cycle de de longueur égale à 4 nécessit $O(n)$ opérations

3.
   - Données : Un graphe $G = (V, E)$, deux sommets u et v distincts de G et un entier k.
   - Question : Existe-t-il un simple chemin entre u et v de longueur inférieure ou égale à k ?

   Le problème 3 est dans NP car étant donnée une suite de sommets P, on peut vérifier en temps polynomial si P est un simple chemin entre u et v de longeur inférieur ou égale à k ($O(n)$ opérations).
   Le problème 3 est dans P. Considérons l'algorithme suivant. Il suffit simplement de calculer le plus court chemin entre u et v . Si la longueur du chemin est plus grande que k, ou si il n'existe pas de chemin entre u et v alors répondre non sinon répondre oui.

$\frac{n}{2}$-CLIQUE est NP(-Complet)

Le language $\frac{n}{2}$CLIQUE est constitué des graphes G qui contiennent une clique de taille au moins $\mid V(G) \mid /2$.

$$\frac{n}{2} - CLIQUE := \left\{ \langle G \rangle \mid G \; a \; une \; \frac{|V(G)|}{2} - CLIQUE \right\}$$

$\frac{n}{2} - CLIQUE \in NP$

Un certificat serait la liste des sommets de la $\frac{n}{2}$-clique. Le vérificateur s'assure que la clique contient bien $\frac{n}{2}$ sommets de G et qu'il s'agit d'un sous-graphe complet en $\frac{n}{2} \times (\frac{n}{2} - 1)$ opérations.
Si un graphe n'est pas dans $n/2$-clique, aucun certificat ne satisfera le vérificateur.