Examen du cours de L3: Calculabilité

14 novembre 2013, durée: 3 heures

Tous les documents sont autorisés, les appareils électroniques sont interdits. Les résultats vus en TD, s'ils sont utilisés, doivent être redémontrés. À titre indicatif, au moins 15 points seront attribués au premier exercice.

1

Pour chacun des problèmes suivants, dire s'ils sont décidables ou non. Justifier.

1. **Donnée:** Une machine de Turing M, un mot w, un entier n

   Question: M s'arrête sur w en moins de n étapes de calcul

2. **Donnée:** Une machine de Turing M

   Question: Tous les mots de $L(M)$ sont de longueur paire

3. **Donnée:** Deux machines de Turing M_1, M_2

   Question: Pour tout mot w, M_1 s'arrête sur w si et seulement si M_2 s'arrête sur w.

4. **Donnée:** une machine de Turing M

   Question: Il existe une machine M' telle que M' calcule en temps linéaire et $L(M) = L(M')$.

5. **Donnée:** une fonction récursive primitive à un argument f

   Question: il existe un entier n tel que $f(n) = 0$

6. **Donnée:** Une machine de Turing M

   Question: M, sur la donnée vide, repasse au moins une fois dans l'état initial

7. **Donnée:** Deux grammaires linéaires G_1, G_2

   Question: $L(G_1) \cap L(G_2) \neq \emptyset$

   Une grammaire (HC) est donnée par un alphabet terminal Σ, un alphabet non-terminal N, un symbole de départ $S \in N$ et un ensemble fini de règles de grammaire de la forme $A \to y$ où $y \in (\Sigma \cup N)^*$. Une *dérivation* de w dans la grammaire est une séquence $x_1 \to \ldots \to x_n = w$ où $x_1 = S$ et, pour tout $i \leq n-1$, il existe une règle de grammaire $A \to y$ et des mots $w_1, w_2 \in (\Sigma \cup N)^*$ tels que $x_i = w_1 A w_2$ et $x_{i+1} = w_1 y w_2$. Le *langage engendré* $L(G)$ par la grammaire G est l'ensemble des mots w de Σ^* tels qu'il existe une dérivation de w dans la grammaire.

Une grammaire est *linéaire* si toutes ses règles sont de la forme $A \to y_1 B y_2$ où $B \in N \cup \{\epsilon\}$ et $y_1, y_2 \in \Sigma^*$.

Ind: on pourra considérer le langage $L = \{w\widetilde{w} \mid w \in \Sigma^*\}$ où $\widetilde{w}$ est l'image miroire du mot w.

8. **Donnée:** une machine de Turing M (à un ruban)

   Question: Il existe un entier k tel que, pour tout mot w, M s'arrête après au plus $k \times |w|^2$ étapes de calcul

   (Question hors barême: même question en remplaçant $|w|^2$ par $|w| \log |w|$).

2

Parmi les problèmes 1,2,3 de la question précédente, dire (en le justifiant) ceux pour lesquels {Donnée | La Question est satisfaite} est récursivement énumérable.

3

On admet l'existence d'une application surjective t des entiers dans les fonctions récursives primitives à un argument (une numérotation des fonctions récursives primitives à un argument). On admet également l'existence d'une fonction récursive totale universelle Eval à deux arguments telle que $\mathsf{Eval}(x, y) = t(x)(y)$.

Montrer que Eval n'est pas récursive primitive.

Solution

1

1. Ce problème est décidable: il suffit de simuler n étapes de calcul de M sur w. Autrement dit, la machine universelle est modifiée (en une machine M') pour commencer par transférer n sur un troisième ruban, puis, après simulation de chaque transition de M, décrémenter le compteur du troisième ruban. Si M arrive dans un état **accept** ou **reject**, M' accepte. Si le compteur atteint 0, M' rejette.
   Points =2, Taux de réussite = 98%

2. Ce problème est indécidable d'après le théorème de Rice: la propriété que tous les mots de L sont de longueur paire est une propriété non-triviale puisque le langage vide est dedans, alors que le langage réduit à un singleton n'y est pas.
   Points=2, Taux de réussite = 85%

3. C'est indécidable, par réduction du problème de l'arrêt : Si M, w est la donnée du problème de l'arrêt, on choisit pour M_1 une machine qui ignore son entrée et simule M sur w et pour M_2 une machine qui ignore son entrée et s'arrête. M s'arrête sur w ssi M_1 s'arrête sur toute entrée ssi (M_1 s'arrête sur toute entrée ssi M_2 s'arrête sur toute entrée).
   Points=2, Taux de réussite = 78%

4. C'est indécidable: on utilise le théorème de Rice. Il suffit de montrer que la propriété de $L(M)$ est non-triviale: Le langage vide a la propriété (choisir M' qui s'arrête immédiatement en rejetant) et le langage universel ne l'a pas, puisqu'il n'est pas récursif.
   Taux de réussite = 45%

5. C'est indécidable: on réduit le problème de l'arrêt. Soit M, w une instance du problème de l'arrêt. Comme vu en cours, il existe une fonction primitive récursive qui, à un entier n associe le codage de la nième configuration du calcul de M sur w et 0 si la machine M s'arrête en moins de n étapes. Il existe un entier n tel que $f(n) = 0$ ssi M s'arrête sur w.
   Points=2, Taux de réussite = 32%

6. On réduit le problème de l'arrêt: si M, w est une donnée du problème de l'arrêt, on construit M' qui simule M et, si M va s'arrêter sur w, revient dans l'état q_0. Par ailleurs, on ajoute un état q_0' à M' et on remplace q_0 par q_0' dans toutes les transitions de M', sauf la transition $\delta(q_0, \$)$ et la transition qui fait revenir dans l'état initial après que la simulation de M sur w soit terminée. De cette façon M' revient dans l'état q_0 ssi M s'arrête sur w.
   Points=2, Taux de réussite = 54%

7. On code le problème de correspondance de Post. Si $(u_1, \dots, u_n)$, $(v_1, \dots, v_n)$ est une instance de PCP, on associe les deux grammaires:
   - G_1: $S \to u_i\widetilde{v_i}$, $S \to u_iS\widetilde{v_i}$ où $\widetilde{v_i}$ est l'image miroir de v_i, pour tout i
   - G_2: $S \to \epsilon$, $S \to aSa$ pour toute lettre a de l'alphabet

$L(G_2)$ est le langage des mots $\{w\widetilde{w} \mid w \in \Sigma^*\}$.

$L(G_1) = \{u_{i_1} \cdots u_{i_k} v_{i_k} \cdots v_{i_1} \mid k \in \mathbb{N}, i_1, \ldots, i_k \in \{1, \ldots, n\}\}$. Par récurrence sur k il existe une dérivation $S \rightarrow^*_{G_1} u_{i_1} \cdots u_{i_k} \cdot S \cdot v_{i_k} \cdots v_{i_1}$ et réciproquement, toute dérivation est de cette forme, par récurrence sur la longueur de la dérivation.

Donc $L(G_1) \cap L(G_2) = \emptyset$ ssi PCP a une solution.
Points=2, Taux de réussite = 6%

8. **Note: la solution ci-dessous correspond au cas où la donnée est une machine qui s'arrête toujours (ce qui était l'intention de l'énoncé), mais les solutions (plus simples) correspondant à l'énoncé tel qu'il a été distrubé sont aussi valabels**

   C'est indécidable: on réduit le complémentaire du problème de l'arrêt. Soit M, w une donnée du problème de l'arrêt. Soit M_1 la machine qui, étant donné un entier n (en base 1), simule n étapes du calcul de M sur w et, si M est sur le point de s'arrêter, écrit n^3 sur le ruban. Si après n étapes de calcul M ne s'est pas arrêtée, alors M_1 s'arrête (sur la donnée n).

   M_1 est par exemple une machine à deux rubans, n étant écrit en unaire sur le premier ruban et la configuration de M sur le deuxième ruban. À chaque étape, M_1 avance sur le premier ruban (tant qu'elle ne lit pas un blanc) et simule M sur le deuxième ruban. M_1 peut être simulée par une machine M_2 à un seul ruban, dont le nombre d'étapes de calcul est majoré par $c \times t_{M_1}(n)^2$ où c est une constante (qui dépend de M, w) et $t_{M_1}(n)$ est le nombre d'étapes de calcul de M_1 sur la donnée n.

   Par construction, le nombre d'étapes de calcul de M_2 sur n est majoré par $k \times n^2 + 1$ si M ne s'arrête pas sur w en moins de n étapes et sinon il est minoré par n^3 et majoré par $k \times n^6 + 1$.

   Il en résulte que M ne s'arrête pas sur w ssi M_2 calcule en temps $k \times n^2 + C$.

   Pour l'extension à $O(|w| \times \log |w|)$: sur la donnée n (en base 1), la machine M_2 commence par marquer $\log n$ sur le ruban. Ceci est possible en temps $O(n \log n)$: supposant n codé par 1^n, on répète, jusqu'à avoir épuisé les 1: remplacer un 1 (ou $1'$) sur deux par 0 et marquer remplacer la première lettre non primée (0 ou 1) par la lettre primée correspondante ($0'$ ou $1'$). On passe $\log n$ fois sur l'entrée (donc temps de calcul $O(n \log n)$) et à l'issue du calcul les $\log n$ premières lettres sont primées.

   Ensuite, M_2 simule M sur w, dans l'espace alloué par les lettres primées: si M s'arrête sur w en moins de $\log n$ étapes, alors M_2 entre dans un long calcul. Sinon, M_2 s'arrête. Cette dernière étape de simulation requiert $O(\log^2 n)$ étapes de calcul (si M s'arrête sur w).

   Points=2, Taux de réussite = 52%

2

1. Tout problème décidable est récursivement énumérable. Donc 1. est récursivement énumérable.
   Points= 0.5, Taux de réussite = 81%

2. Ce problème n'est pas récursivement énumérable. Comme tout langage récursivement énumérable et co-récursivement énumérable est récursif, d'après la question 1.2, il suffit de montrer que le langage de cette question est co-récursivement énumérable. Pour cela, la machine M_0 réalise les opérations suivantes:

   Pour $n := 0$ à $+\infty$ faire

       Pour tous les mots w de longueur impaire et de longueur inférieure à n faire

           Simuler n étapes de M sur w

           Si M va s'arrêter sur w, alors s'échapper de toutes les boucles et accepter.

   Si M_0 accepte $< M >$, alors, par construction, M accepte au moins un mot de longueur impaire.

   Réciproquement, si M accepte un mot w de longueur $2k+1$, alors M_0 s'arrête en acceptant, car la boucle externe ne sera éxécutée qu'au plus jusqu'à $n = \max(2k+1, t)$ où t est le temps de calcul de M sur w. (Et la boucle interne termine toujours).

   Donc M_0 accepte $< M >$ ssi M accepte au moins un mot de longueur impaire. Ce qui montre que le langage de la question 1.2 est co-récursivement énumérable et donc pas récursivement énumérable.
   Points=1.25, Taux de réussite=26%

3. Ce problème n'est pas récursivement énumérable: on réduit le problème du non-arrêt de M sur w.

   On considère une machine M_1 qui ne s'arrête sur aucune entrée et une machine M_2 qui ignore son entrée et simule M sur w.
   (M ne s'arrête pas sur w) ssi (pour tout x, M_2 ne s'arrête pas sur x) ssi (pour tout x, M_1 s'arrête sur x ssi M_2 s'arrête sur x)

   Or le problème du non-arrêt n'est pas récursivement énumérable, car le problème de l'arrêt est récursivement énumérable et pas récursif. Il en résulte que le le langage de la question 1.3 n'est pas récursivement énumérable.
   Points=1.25, Taux de réussite =9%

3

Si Eval était récursive primitive, alors il existerait un entier n tel que $t(n)(x) = \mathsf{Eval}(x, x) + 1$. Mais alors $\mathsf{Eval}(n, n) = t(n)(n) = \mathsf{Eval}(n, n) + 1$, ce qui est absurde.
Points= 2.5, Taux de réussite= 33%