Devoir de Calculabilité

À rendre le 10 octobre 2016 au plus tard

Dans tout l'énoncé, les machines de Turing sont supposées disposer de deux rubans, le deuxième ruban étant dit "ruban de sortie" (même si aucune hypothèse n'est effectuée sur la lecture/écriture sur ce ruban). Dans ce contexte, la donnée de la machine de Turing sera toujours sur le premier ruban: la configuration initiale de M sur w est $(q_0, (\epsilon, \$w), (\epsilon, \$)$. Le résultat du calcul de M sur w (si M s'arrête sur w) sera le contenu du deuxième ruban (auquel on enlève le préfixe \$ et les blancs en fin de mot).

Question 1 (un théorème de point fixe)

On note M_U la machine universelle.

Soit f est une fonction calculable qui envoie les codes de machines de Turing sur des codes de machine de Turing (et renvoie le mot vide sur les entrées qui ne sont pas des codes de machines de Turing).

Montrer qu'il existe une machine M telle que, sur toute donnée x,

$$M_U(f(< M >), < x >) =< M(x) > .$$

Ind: considérer la fonction g qui à $< M >$ associe le code de la machine $M(M)$ qui à x associe $M(< M >)(x)$ quand $M(< M >)$ est le code d'une machine de Turing et ne termine pas sinon. Considérer ensuite $g(< M_{f \circ g} >)$.

Question 2

Dans cette partie, on appellera *virus* une machine de Turing M comportant deux rubans et un état spécial q_{emit}, telle que, sur la donnée ϵ, le calcul de M passe par une configuration $(q_{\text{emit}}, (w_1, w_2), (\epsilon, \$ < M >))$.

1. Montrer qu'il existe des virus. (On pourra utiliser la question 1)
2. Montrer qu'avec cette définition, il est possible de détecter les virus: il existe une machine M_d telle que, sur la donnée $< M >$, M_d s'arrête en acceptant si et seulement si M est un virus.
3. Montrer qu'en revanche l'ensemble des virus n'est pas récursif.

Question 3

L'ensemble $\mathcal{V}$ est *machines de Turing infectées par un virus* est le plus petit ensemble de (codes de) machines de Turing tel que:

- Les virus appartiennent à $\mathcal{V}$
- Si le calcul de M sur ϵ passe par une configuration dans laquelle le ruban de sortie contient le code d'une machine de Turing $M' \in \mathcal{V}$, alors $M \in \mathcal{V}$.

Montrer que $\mathcal{V}$ est récursivement énumérable et pas récursif.

Question 4

Dans cette partie, on appellera *virus* une machine de Turing M comportant un ruban de sortie tel que, sur *toute donnée*, le calcul de M passe par une configuration dans laquelle le ruban de sortie contient $< M >$.

La définition des machines infectées par un virus est la même que dans la question précédente, avec cette nouvelle définition de virus.

Montrer qu'alors l'ensemble des machines infectées par un virus n'est ni récursivement énumérable, ni co-récrusivement énumérable.

Question 5

On revient à la définition de virus de la question 2. Mais il est plus réaliste de considérer qu'un virus ne se reproduit pas nécessairement à l'identique, mais produit un clone qui est fonctionnellement identique.

Soit $\sim$ l'équivalence observationelle: étant données deux machines de Turing V, V', $V \sim V'$ si, pour toute donnée x la séquence $w_1, \ldots, w_n, \ldots$ des contenus du deuxième ruban quand V est dans l'état q_{emit} est la même que la séquence $w'_1, \ldots, w'_n$ des contenus du deuxième ruban quand V' est dans l'état q_{emit}.

Montrer que l'ensemble des machines M qui sont équivalentes à un virus n'est ni récursivement énumérable, ni co-récursivement énumérable. Autrement dit, ce type de virus est indétectable.

Solution

Question 1

Montrons que g est calculable: sur une entré w, la fonction g est calculé par:

- Vérifie que w est le code d'une machine de Turing.
  - Si oui, retourne $< M_w >$ ou M_w est la machine qui sur une entré x:
    - Simule w sur w.
    - Si termine, vérifie que le résultat r code une machine de Turing: si oui simule r sur x, sinon boucle.
  - Sinon, retourne le code d'une machine qui ne termine jamais.

Soit M_g calculant g et M_f calculant f. On peut construire $M_{f\circ g}$ calculant $f \circ g$. On remarque que g envoie les codes de MT sur des codes de MT, donc soit $< M >= g(< M_{f\circ g} >)$. Finallement:

$$\begin{aligned} M_u(f(< M >), < x >) &= M_u((f \circ g)(< M_{f\circ g} >), < x >) \\ &= M_u(M_{f\circ g}(< M_{f\circ g} >), < x >) \\ &= M_u(g(< M_{f\circ g} >), < x >) \qquad (M_{f\circ g}(< M_{f\circ g} >) \text{ code une MT}) \\ &= M_u(< M >, < x >) \\ &=< M(x) > \end{aligned}$$

Question 2

1. On considère la fonction f qui, au code d'une machine de Turing M associe le code de la machine W_M qui, sur toute entrée, efface son entrée, écrit $< M >$ sur la bande de sortie puis s'arrête. Cette fonction f est récursive, donc, d'après la question 1, il existe une machine M_f telle que pour tout x:

   $$M_U(f(< M_f >), < x >) =< M_f(x) >$$

   Or $f(< M_f >) =< W_{M_f} >$ et donc

   $$M_U(f(< M_f >), < x >) =< W_{M_f}(x) >=<< M_f >>$$

   On a donc, pour tout x, $M_f(x) =< M_f >$. Pour conclure, il suffit de remarquer que on peut supposer que la fonction g de la question 1 retourne le code d'une machine qui efface sa bande d'entré avant de s'arrêter. Dans ce cas, le point fixe M_f termine avec sa bande d'entré vide. En choisissant $q_{emit} =$ **accept**, on obtient le résultat souhaité.

2. M_d, sur la donnée $< M >$, exécute M sur le mot vide (on utilise la machine universelle) et teste, après chaque étape, si $< M >$ est présent sur le ruban de sortie. Si c'est le cas, elle s'arrête en **accept**. Si la simulation de M sur ϵ termine alors M_d s'arrête en **reject**.

3. On réduit le problème de l'arrêt sur le mot vide: soit $< M >$ le code d'une machine de Turing.

   Soit f la fonction qui au code d'une machine M_1 associe le code de la machine qui éxécute M sur le mot vide puis qui, si M s'arrête, écrit le code de M_1 sur la bande de sortie et s'arrête.

   f est récursive donc, d'après la question 1, il existe une machine M_f (dont le code est effectivement constructible !) telle que $M_U(f(< M_f >), < x >) =< M_f(x) >$. $M_U(f(< M_f >, \epsilon)$ exécute M sur le mot vide puis, si M s'arrête, écrit le code de M_f sur sa bande de sortie et s'arrête. De plus comme à la question 2.1, on peut supposer que M_f efface sa bande d'entré avant de s'arrêter.

   Donc M_f est un virus ssi M s'arrête sur ϵ.

Question 3

$\mathcal{V}$ n'est pas co-récursivement énumérable: on réduit le problème de l'arrêt: soit V un virus, à une machine M on associe la machine M' qui simule M sur le mot vide, puis, en cas d'arrêt, écrit $< V >$ sur la bande de sortie. M' est infectée par un virus ssi M s'arrête sur le mot vide.

$\mathcal{V}$ est récursivement énumérable: la machine DV est consrtuite comme suit:

1. Elle conserve sur son ruban une liste de threads:

   $$< \gamma_1 >, < M_1 >, \ldots, < \gamma_n >, < M_n >$$

   Pour tout i, $< \gamma_i >$ est le code d'une configuration de la machine de Turing M_i.

2. Elle exécute, pour chacun des éléments de la liste une étape de calcul: on arrive dans une configuration:

   $$< \gamma'_1 >, < M_1 >, \ldots, < \gamma'_n >, < M_n >$$

3. Elle teste si l'un des rubans de sortie de γ'_i contient $< M_i >$. Si c'est le cas, DV s'arrête en succès.

4. Sinon, pour soit $M'_1, \ldots, M'_l$ l'ensemble des codes de machine de Turing apparaissant sur les rubans de sorties des configurations $< \gamma'_1 >, \ldots, < \gamma'_n >$. On ajoute à la liste les configurations initiales de ces machines sur ϵ, ainsi que les codes de machines correspondants (si M_T est une machine de Turing on note $\gamma^{\epsilon}_{M_T}$ la configuration initial de M_T sur ϵ):

   $$< \gamma'_1 >, < M_1 >, \ldots, < \gamma'_n >, < M_n >, < \gamma^{\epsilon}_{M'_1} >, < M'_1 >, \ldots, < \gamma^{\epsilon}_{M'_l} >, < M'_l >$$

5. On retire de la liste les configurations finales, puis on retourne à l'étape 1.

Initiallement, sur une entré $< M >$, DV rajoute la configuration initial de M sur ϵ (le ruban est de la forme $< \gamma^{\epsilon}_M >, < M >$).

Correction et Terminaison: Soit $(V_n)_{n\geq 0}$ les ensembles de machine de Turing défine par V_0 est l'ensemble des virus et pour tout $n > 0$, V_n est l'ensemble des machine M tel que:

- Les virus appartiennent à $\mathcal{V}$
- Si le calcul de M sur ϵ passe par une configuration dans laquelle le ruban de sortie contient le code d'une machine de Turing $M' \in V_{n-1}$, alors $M \in V_n$.

Les (V_n) sont une suites croissantes d'ensembles, et $\mathcal{V} = \cup_{i\geq 0} V_n$. Par conséquent $< M >$ est une machine infecté ssi il existe n tel que $< M > \in V_n$, ce qui implique que M est un virus ou qu'il existe $M' \in V_{n-1}$ tel que M exécuté sur ϵ écrive le code de M'. On montre alors par induction sur n que si $< M > \in V_n$ alors DV termine sur $< M >$ et accepte.

Réciproquement une machine M est accepté par DV si il existe une suite $(M_i)_{1\leq i\leq k}$ tel que pour tout i, M_i écrit le code de M_{i+1} lorsque elle est exécuté sur ϵ, $M_1 = M$ et M_k est un virus. On en déduit que $M \in V_k$.

Question 4

Les virus V de la question 2 ne dépendent pas de l'entrée. On réduit donc le problème de l'arrêt sur le mot vide, exactement comme dans la question précédente.

$\mathcal{V}$ n'est pas récursivement énumérable: le problème, étant donné $< M >$ est ce que M s'arrête sur toute donnée ? (arrêt universel) n'est pas récursivement énumérable. En effet, on réduit le problème du non arrêt comme suit: à $< M, w >$ on associe la machine M' qui, sur la donnée x effectue $|x|$ étapes de M sur w: si M s'arrête en moins de $|x|$ étapes sur w, alors M' boucle. Sinon, M' s'arrête. Alors, M' s'arrête sur toute entrée ssi M ne s'arrête pas sur w.

On réduit ensuite l'arrêt universel à $\mathcal{V}$: à une machine M on associe la machine M' qui, sur la donnée x exécute M sur x et, en cas d'arrêt, exécute ensuite un virus V (sur la donnée vide). M s'arrête sur toutes les entrées ssi M' est infectée par un virus.

Question 5

Montrons que ce problème n'est pas récursivement énumérable en réduisant le probléme de l'arrêt universelle (le fait qu'il n'est pas coRE est trivial).

Soit f la fonction calculable qui a un code de machine de Turing $< M >$ associe le code $< M' >$ ou M' est la machine qui sur toute entrée x:

- efface sa bande d'entrée et de sortie.
- écrit $< M >$ sur sa bande de sortie et passe dans un état q_{emit}.
- recommence depuis le début.

En utilisant le théorème de point fixe sur la fonction f, on construit une machine V qui écrit infiniment souvent son propre code sur le ruban de sortie. Soit x_n est une énumération des mots, et soit M' qui exécute M successivement sur les x_i et, après chacune des exécutions, écrit le code de V sur la bande de sortie.

M' est équivalent à un virus ssi M' est équivalent à V ssi l'exécution de M' ne termine pas ssi pour tout x, M termine sur x.