Incalculabilité

Exercice 1. *Dénombrable ou non ?*

Pour chacun des ensembles suivants, donner trois exemples d'éléments de l'ensemble et déterminer si l'ensemble est dénombrable ou non :

1. $\mathbb{N} \times \{0, 1, a\}$;
2. $\mathbb{N} \times \mathbb{R}$;
3. $\mathbb{Q}[\sqrt{2}] = \{a + b\sqrt{2} : a, b \in \mathbb{Q}\}$;
4. l'ensemble $\{0, 1\}^\omega$ des chaînes infinies de bits ;
5. l'ensemble des sous-ensembles finis de $\mathbb{N}$;
6. l'ensemble des sous-ensembles de $\mathbb{N}$;
7. l'ensemble des expressions régulières.

Exercice 2. *Constructions*

1. **i.** Soit X et Y deux ensembles infinis dénombrables. Montrer que $X \times Y$ est dénombrable.
   ii. Soit $X_0, X_1, \ldots$ des ensembles infinis dénombrables. Est-ce que $\prod_{i=0}^{\infty} X_i$ est dénombrable ?
2. **i.** Soit X et Y deux ensembles infinis dénombrables. Montrer que $X \cup Y$ est dénombrable.
   ii. Soit $X_0, X_1, \ldots$ des ensembles infinis dénombrables. Est-ce que $\bigcup_{i=0}^{\infty} X_i$ est dénombrable ?
   iii. Construire un ensemble indénombrable d'ensembles dénombrables, dont l'union soit indénombrable.
   iv. Construire un ensemble indénombrable d'ensembles dénombrables, dont l'union soit dénombrable.

Exercice 3. *Nécessité des fonctions partielles*

L'entreprise Disrupt® propose son super nouveau langage *Halt!* avec la promesse qu'aucun programme écrit dans son langage ne peut rentrer dans une boucle infinie. Bien sûr, Disrupt® promet également qu'on peut programmer *tout ce qu'on veut* dans son langage, tant que ça ne boucle pas. C'est vraiment fantastique…

1. **i.** Traduire la promesse « aucun programme ne peut rentrer dans une boucle infinie » en une propriété sur les fonctions calculées par les programmes écrits en *Halt!*.
   ii. Justifier, intuitivement, que si on sait calculer $f(n)$ pour $f : \mathbb{N} \to \mathbb{N}$ avec *Halt!*, on sait aussi calculer $f(n) + 1$.
   iii. Démontrer que l'ensemble des fonctions qu'on peut programmer avec *Halt!* est dénombrable.
2. Montrer que Disrupt® fait de la publicité mensongère : les trois propriétés de la question précédente impliquent qu'il existe une fonction *g intuitivement calculable* qu'on ne peut pas programmer avec *Halt!*. *Indication. Utiliser une preuve par diagonalisation.*
3. Dans la preuve précédente, qu'est-ce qui ne marche pas si le langage permet des fonctions qui bouclent ? Autrement dit, pourquoi cela ne démontre pas que pour tout langage, il existe un algorithme *intuitivement calculable* non programmable dans le langage ?

Exercice 4. *Décidable ou non ?*

Déterminer si chacun des problèmes suivants est décidable.

1. Étant donné la fonction de transition d'un automate cellulaire et une configuration initiale (avec un nombre fini de cases non blanches), est-ce que l'automate atteint une configuration *stable*, qui n'évolue plus ?
2. Étant donné un automate fini déterministe A et un mot w, est-ce que A accepte w ?
3. Étant donné un automate fini déterministe A, est-ce que le langage accepté par A est infini ?
4. Étant donné un algorithme A, est-ce que $L(A) = \emptyset$?
5. Étant donné un algorithme A, est-ce que pour tout $w \in L(A)$, $\overline{w} \in L(A)$? *Le mot $\overline{w}$ est le complémentaire de w obtenu en inversant chaque bit.*
6. Étant donné deux algorithmes A et B, est-ce que $A(w) = B(w)$ pour toute entrée (c'est-à-dire en particulier $A(w) \uparrow \iff B(w) \uparrow$) ?
7. Étant donné une machine de Turing $\mathcal{M}$ et une entrée w, est-ce que $\mathcal{M}$ modifie la partie du ruban qui contient w ?

8. Étant donné un programme dans votre langage préféré, est-ce que le programme affiche `J'adore la calculabilité.` ?

On définit la fonction *busy beaver* $\text{BB} : \mathbb{N} \to \mathbb{N}$ comme suit : pour un entier k, on considère toutes les machines de Turing à k états, qui terminent sur l'entrée vide ; soit $\text{BB}(k)$ le nombre maximal d'étapes nécessaires pour une de ces machines pour atteindre sa configuration d'arrêt. Autrement dit, toutes celles qui s'arrêtent le font en $\leq k$ étapes de calcul, et l'une le fait en exactement k étapes de calcul.

9. Montrer que la fonction BB n'est pas calculable.

Exercice 5. *Un peu de théorie*

1. Montrer que L est décidable si et seulement si $\overline{L}$ est décidable.
2. Montrer que si un langage L est reconnaissable et si $L \leq_m \overline{L}$, alors L est décidable.
3. **i.** Montrer que L est reconnaissable si et seulement s'il est le *domaine de définition* d'un algorithme : il existe un algorithme A tel que $A(w)\downarrow$ si et seulement si $w \in L$.
ii. Montrer que L est reconnaissable si et seulement s'il est l'*image* d'un algorithme : il existe un algorithme A tel que $L = \{A(x) : x \in \{0,1\}^*\}$.
4. Montrer que L est reconnaissable si et seulement si $L \leq_m K$ où K est le problème de l'arrêt.
5. Soit D un langage décidable, non vide et $\neq \{0,1\}^*$. Montrer que L est décidable si et seulement si $L \leq_m D$.
6. Soit L un langage reconnaissable. Montrer que si L contient exactement un mot de chaque longueur, alors L est décidable.
7. Soit L un langage énumérable par un algorithme E qui énumère les mots de L par ordre de taille, puis lexicographique (pour ceux de même taille). Montrer que L est décidable.

Exercice 6. *Calculabilité relative*

On imagine posséder une instruction magique pour résoudre le problème de l'arrêt : étant donné $\langle A\rangle$ et w, $\text{STOP?}(\langle A\rangle, w)$ renvoie 1 si $A(w)\downarrow$ et 0 si $A(w)\uparrow$. Cette instruction magique s'appelle l'*oracle* K (le problème de l'arrêt) et un algorithme qui l'utilise est un *algorithme avec oracle* K, noté A^K. On peut, comme pour un algorithme sans oracle, représenter A^K par un mot binaire $\langle A^K\rangle$.

1. Montrer que déterminer, étant donné $\langle A^K\rangle$ et un mot w, si $A^K(w)\downarrow$ est indécidable.
2. Le problème est-il décidable par des algorithmes à oracle ? Autrement dit, existe-t-il un algorithme à oracle H^K tel que $H^K(\langle A^K, w\rangle) = 1$ si $A^K(w)\downarrow$ et 0 si $A^K(w)\uparrow$?

Exercice 7. *Autres problèmes indécidables*

1. Déterminer si le problème de correspondance de Post est décidable dans les cas suivants :

i. lorsque chaque domino possède deux mots de même longueur ;
ii. lorsque l'alphabet utilisé ne possède qu'une seule lettre ;
iii. lorsque l'alphabet utilisé possède deux lettres.

2. Le problème du pavage est-il décidable en dimension 1 ? *Précisions : on a en entrée un ensemble de tuiles possédant une couleur à gauche et une à droite, et on cherche à remplir une ligne complète en respectant les couleurs, étant donné une tuile de départ.*