TD Introduction à la Calculabilité

Partie 1 : machines de Turing

Exercice 1. Donner des machines de Turing qui décident (donc qui s'arrêtent toujours) les langages suivants :

L1 = { wcw| $w \in \{a,b\}^*$} L2 = {ww| $w \in \{a,b\}^*$}

Exercice 2. Définir les machines de Turing correspondant à des automates finis déterministes. Autrement dit donner, pour un AFD A = (Q, V, q0, F, δ) une MT m = (Q', Γ, V, B, q0, F', δ') telle que L(m)=L(A).

Exercice 3. Donner des machines de Turing (à un ou plusieurs rubans) qui réalisent les fonctions M et P définies comme suit :

x M y = si $x > y$ alors $x - y$ sinon 0 (avec x, y et le résultat codés en unaire)
x P y = $x + y$ (avec x, y et le résultat codés en binaire)

Partie 2 : fonctions calculables et ensembles récursivement énumérables

Exercice 4. Montrer que l'inverse d'une fonction $f : \mathbb{N} \to \mathbb{N}$, calculable et bijective, est calculable. Que peut-on dire si f est surjective mais pas injective, ou vice-versa ?

Exercice 5.

1. Montrer que R est fermé par union, intersection, complémentaire.
2. Montrer que RE est fermé par union et intersection.
3. Montrer que si E et son complémentaire sont dans RE, alors ils sont dans R.

Rappel : un ensemble E est dans R ssi sa fonction caractéristique (qui vaut 1 si x appartient à E, 0 sinon) est totale calculable. E est dans RE ssi E est le domaine d'une fonction partielle calculable.

Exercice 6. Soit L un langage de programmation qui ne permet de programmer que des fonctions totales calculables. Montrer que L est incomplet, c'est-à-dire qu'il existe forcément des fonctions totales calculables qui ne peuvent pas être programmées dans L.

Rappel : une fonction est totale calculable ssi elle est calculable par une machine de Turing qui s'arrête toujours.

Exercice 7.
Soit E_n = {x | x est le résultat d'une MT d'alphabet {0,1,B} ayant $n+1$ états, sur l'entrée n}.
Montrez que E_n est fini et non vide.

Soit g la fonction suivante :
$g : \mathbb{N} \to \mathbb{N}$ telle que $g(n)$ = Max(E_n)
Montrez que g n'est pas calculable.

Exercice 8. Soit h la fonction suivante :
$h : \mathbb{N} \to \mathbb{N}$ telle que $h(n)$ = si MTU(n,n) s'arrête alors 1 sinon 0
h est-elle calculable ?

Exercice 9. Donner un ensemble non récursivement énumérable.

Indication : Soit H = {n | $h(n) = 1$} où h est la fonction définie dans l'exercice 8. Montrer que H n'est pas récursif, mais qu'il est récursivement énumérable. En déduire que le complémentaire de H n'est pas récursivement énumérable.

Partie 3 : indécidabilité

Exercice 10. Soient G_1 et G_2 deux grammaires LL(1). Montrer que le problème « $L(G_1) \bigcap L(G_2) \neq \emptyset$ » est indécidable (en réduisant le problème de Post à ce problème). En déduire que l'on ne peut pas décider si une grammaire hors-contexte donnée est ambiguë.

Exercice 11. (*Théorème de Rice pour les langages récursivement énumérables*) Montrer que toute propriété des langages r.e., non triviale (i.e. qui n'est ni toujours vraie, ni toujours fausse), est indécidable. On peut formaliser le problème de la manière suivante :

Soit L = {<*m*> | P(*m*)} où P est un prédicat sur les machines de Turing qui ne dépend que du langage reconnu par *m*, et qui n'est ni toujours vrai, ni toujours faux. On note <*m*> un codage de la machine *m*. Montrer que L n'est pas récursif.

Indication : On définit le langage H = {<*m*>#*w* | *m* s'arrête sur *w*}. Montrer que si L est récursif alors H est récursif. Conclure en remarquant que H n'est pas récursif.

Rappel : les langages réguliers, hors-contexte et sous-contexte sont récursifs (il existe des algorithmes de décision).

Exercice 12. Soit

1. L_w = {<*m*> | *w* ∈ L(*m*)} (pour *w* quelconque)
2. L_{ne} = {<*m*> | L(*m*) ≠ ∅ }
3. L_e = {<*m*> | L(*m*) = ∅ }
4. L_k = {<*m*> | |L(*m*)| ≥ *k*} (pour *k* quelconque ∈ ℕ)
5. L_f = {<*m*> | L(*m*) est fini}
6. L_{reg} = {<*m*> | L(*m*) est régulier}
7. L_{hc} = {<*m*> | L(*m*) est hors-contexte}
8. L_r = {<*m*> | L(*m*) est récursif}
9. L_{nr} = {<*m*> | L(*m*) n'est pas récursif}

Montrez les assertions suivantes :

1. Aucun des langages ci-dessus n'est récursif.
2. L_w est r.e. (quel que soit *w*)
3. L_{ne} est r.e.
4. L_e n'est pas r.e.
5. L_k est r.e. (quel que soit *k*)
6. L_r n'est pas r.e.
7. L_{nr} n'est pas r.e.

Exercice 13. Soient L et L' deux langages r.e. Le problème « L = L' » est-il décidable ? Qu'en est-il du problème « L ⊆ L' » ?

Exercice 14. (*Théorème de Rice pour les programmes*)

On appelle *propriété de correction* d'un programme *X* une propriété qui ne dépend que de la fonction calculée par le programme. *Exemples* : *X* calcule le pgcd de 2 nombres, *X* s'arrête pour toute entrée, *X* s'arrête pour au moins une entrée, *X*(42) vaut 27, *X* calcule la même fonction que *Y*. *Contre-exemple* : *X* a plus de 42 mots réservés… Le théorème de Rice pour les programmes énonce que toute propriété de correction non triviale (non toujours vraie ou toujours fausse) pour les programmes d'un langage de programmation usuel est indécidable.

Montrer que toute propriété P des fonctions calculables qui n'est pas triviale est indécidable. On s'inspirera de la preuve du théorème de Rice pour les langages r.e.