Logique et théorie du calcul

Quelques exercices de calculabilité et complexité

Exercice 1 – Fonction d'Ackerman

La fonction d'Ackerman est la fonction $A : \mathbb{N}^2 \rightarrow \mathbb{N}$ telle que

$$\begin{aligned}A(0, n) &:= n + 1 \\A(m + 1, 0) &:= A(m, 1) \\A(m + 1, n + 1) &:= A(m, A(m + 1, n))\end{aligned}$$

1. Montrer que, pour tout $m \in \mathbb{N}$, la fonction $A_m : n \mapsto A(m, n)$ est récursive primitive.
2. Montrer que A est strictement croissante en ses deux variables.
3. Montrer que pour toute fonction récursive primitive $f : \mathbb{N}^k \rightarrow \mathbb{N}$ il existe un entier m tel que A_m majore f , c'est-à-dire $\forall x_1, \dots, x_k \in \mathbb{N}, f(x_1, \dots, x_k) \leq A(m, \max_{i \leq k} x_i)$.
4. En déduire que A n'est pas récursive primitive.

Exercice 2 – Castor affairé

On considère des machines de Turing déterministes à un unique ruban (qui sert donc d'entrée, de travail et de sortie) utilisant l'alphabet $\{0, 1\}$. Pour tout entier n , on appelle $c(n)$ le plus grand entier ℓ tel qu'il existe une machine à n états (sans compter l'état final) qui, en partant d'un ruban vide, termine en ayant écrit le mot 1^ℓ (donc ℓ fois le caractère 1).

1. Justifier que c est bien définie et qu'elle est strictement croissante.
2. Donner la meilleure minoration que vous pouvez pour $c(1)$, $c(2)$ et $c(3)$.
3. Montrer que pour toute fonction calculable $f : \mathbb{N} \rightarrow \mathbb{N}$ il existe un entier n_0 tel que pour tout $n > n_0$ on a $c(n) > f(n)$.
4. En déduire que c n'est pas calculable.

Exercice 3 – Oubliosité

On considère des machines de Turing à un unique ruban (qui sert donc d'entrée, de travail et de sortie). Une machine de Turing est dite *oublieuse* si, sur une entrée w , la position de ses têtes à la i -ème étape de calcul ne dépend que de la longueur de w et de i (et pas des lettres de w , donc). Montrer que pour tout langage L décidable en temps $O(f(n))$, il existe une machine de Turing oublieuse qui décide L en temps $O(f(n)^2)$.

Exercice 4 – Trop petite complexité

On considère ici des machines de Turing où on distingue le ruban d'entrée du ruban travail, en ne tenant compte que du ruban de travail pour déterminer l'espace utilisé. Pour simplifier, on supposera qu'il y a un unique ruban de travail.

1. Donner une caractérisation des langages décidables en temps $O(1)$.
2. Que dire des langages décidables en temps $o(n)$?
3. Montrer qu'un langage est calculable en espace $O(1)$ si et seulement si il est calculable en espace nul.
4. Montrer que si un langage est décidable en espace $o(\log \log n)$, alors il l'est en espace constant (donc nul).

Exercice 5 – NP-complétude

On rappelle les définitions suivantes :

1. Un *littéral* est une formule logique de la forme x ou $\neg x$ pour une certaine variable booléenne x .
2. Une *clause* est une disjonction de littéraux, donc une formule de la forme $\ell_1 \vee \ell_2 \vee \dots \vee \ell_k$ où les ℓ_i sont des littéraux.
3. Une *forme conjonctive* est une conjonction de clauses, donc une formule de la forme $c_1 \wedge c_2 \wedge \dots \wedge c_n$ où les c_j sont des clauses.

Le problème SAT consiste à déterminer si, pour une forme conjonctive F donnée, il existe une valuation des variables booléennes telle que F soit vraie. Le théorème de Cook montre que ce problème est NP-complet.

Pour un entier k , on appelle k -SAT la restriction de ce problème au cas de formes conjonctives où chaque clause a au plus k littéraux.

4. Justifier que si SAT est décidable en temps polynomial, alors 3-SAT aussi.
5. Soient c et d deux clauses. Soit x une variable booléenne qui n'apparaît pas dans c et d . Montrer que $c \vee d$ est satisfiable si et seulement si $(c \vee x) \wedge (d \vee \neg x)$ est satisfaisable.
6. En déduire qu'il existe une réduction de SAT dans 3-SAT calculable en temps polynomial, et donc que 3-SAT est NP-complet.
7. Montrer que 2-SAT est décidable en temps polynomial.