I Décidabilité

On rappelle que le problème de l'arrêt est indécidable :

ARRET
Instance : le code source d'un programme f et un argument x
Question : f termine-t-il sur l'entrée x ?

Pour chacun des problèmes suivants, dire s'il est décidable ou non en le prouvant.

1. **ARRET-N**
   Instance : une fonction f, un argument x et un entier n.
   Question : est-ce que l'exécution de $f(x)$ termine en moins de n étapes ?
2. **ZERO**
   Instance : une fonction f et un argument x.
   Question : est-ce que $f(x)$ renvoie 0 ?
3. **SAC-A-DOS**
   Instance : un ensemble d'objets de poids $p_1, \ldots, p_n$ et de valeurs $v_1, \ldots, v_n$, un poids maximal P et une valeur minimale V.
   Question : existe-t-il un sous-ensemble d'objets de poids total inférieur à P et de valeur totale supérieure à V ?

II Semi-décidabilité

Un problème de décision est dit semi-décidable s'il existe un algorithme qui :

- répond correctement (et en temps fini) pour toutes les instances positives du problème ;
- répond correctement ou ne termine pas pour les instances négatives.

On demande simplement que l'algorithme ne réponde jamais « Oui » pour une instance négative : il peut répondre « Non » pour certaines de ces instances et ne pas terminer pour d'autres.

Pour un problème de décision A, on définit le problème complémentaire de A, noté co-A, comme le problème de décision ayant les mêmes instances que A, mais des réponses opposées. Autrement dit, les instances positives de co-A sont exactement les instances négatives de A, et les instances négatives de co-A sont exactement les instances positives de A.

1. Le problème `ARRET` est-il semi-décidable ?
2. Montrer qu'un problème A est décidable si et seulement si A et co-A sont tous les deux semi-décidables.
3. Le problème co-`ARRET` est-il semi-décidable ?
4. **$\texttt{ARRET}_\forall$**
   Instance : une fonction f
   Question : le calcul de $f(x)$ termine-t-il pour tout x ?
   (a) Ce problème est-il décidable ?
   (b) Est-ce que co-$\texttt{ARRET}_\forall$ est semi-décidable ?
   (c) Est-ce que $\texttt{ARRET}_\forall$ est semi-décidable ?

III Castor affairé

On considère une version idéalisée du langage OCaml où la mémoire est illimitée et le type `int` permet de représenter des entiers arbitrairement grands.

On dit qu'une fonction $f : \mathbb{N} \longrightarrow \mathbb{N}$ est calculable s'il existe une fonction OCaml `f : int -> int`, dont l'exécution termine pour tout argument positif ou nul, qui calcule les images par f.

On appelle programme une expression OCaml de type `int`. Si l'évaluation de cette expression termine (sans erreur), on dit que le programme calcule la valeur de l'expression. On suppose que les programmes OCaml sont écrits en utilisant les 128 caractères ASCII, et l'on appelle taille d'un programme son nombre de caractères.

Par exemple, les deux programmes suivants calculent respectivement les valeurs 13 et 120 :

```
7 + 6
```

```
let rec fact n =
      if n = 0 then 1
      else n * fact (n - 1)
      in
      fact 5
```

alors que le programme ci-dessous ne termine pas, et ne calcule donc pas de valeur :

```
let rec fact n = n * fact (n - 1) in
fact 5
```

Un castor affairé (busy beaver en anglais) est un programme dont l'exécution termine et qui calcule une valeur la plus grande possible parmi tous les programmes de même taille. On définit le problème de décision `CASTOR` :

CASTOR

Instance : le code source d'un programme OCaml P
Question : P est-il un castor affairé ?

Par exemple, le programme suivant :

```
let k=99 in k*k*k
```

est un programme de taille 17, qui termine et renvoie 970299. Ce n'est pas un castor affairé, puisque le programme suivant (qui n'est toujours pas un castor affairé) renvoie une valeur plus grande :

```
99999999999999999
```

Pour un entier $n \geq 0$, on notera $C(n)$ la valeur maximale renvoyée par un programme OCaml de taille n qui termine et renvoie un entier. On pose $C(0) = 0$ par convention.

On cherche à montrer que la fonction C n'est pas calculable, et que le problème `CASTOR` n'est pas décidable.

1. Combien existe-t-il de programmes OCaml à n caractères, en supposant qu'on dispose de 128 caractères différents possibles ? Expliquer pourquoi le fait qu'il y en ait un nombre fini ne permet pas de conclure que la fonction C est calculable.
2. Montrer que la fonction C est correctement définie.
3. Montrer que C est une fonction croissante, puis que pour $n \in \mathbb{N}$, on a $C(n+2) > C(n)$.
4. Que vaut $C(1)$?

On considère $f : \mathbb{N} \longrightarrow \mathbb{N}$ une fonction calculable.

5. Montrer qu'il existe un entier k tel que pour tout $n \in \mathbb{N}$, $f(n) \leq C(\lfloor \log_{10} n \rfloor + k)$.
6. Montrer que C n'est pas calculable.
7. En déduire que `CASTOR` n'est pas décidable.
8. Montrer, en utilisant la non-calculabilité de C, que le problème de l'arrêt est indécidable. On ne demande pas simplement de prouver la non-décidabilité du problème de l'arrêt, comme cela a pu être fait dans le cours, mais bien de la déduire de la non-calculabilité de C.

IV Classes de complexité

On rappelle que P $\subset$ NP $\subset$ EXP $\subset$ Décidables.
Donner la classe de complexité la plus précise possible des problèmes suivants :

1. `REGEXP-EQUIV`
   Instance : deux expressions régulières e_1 et e_2.
   Question : $L(e_1) = L(e_2)$?

2. `CHEMIN-`$\leq$
   Instance : un graphe $G = (S, A)$, deux sommets $s, t \in S$ et un entier k.
   Question : existe-t-il un chemin élémentaire de s à t de longueur $\leq k$?

3. `CHEMIN-`$\geq$
   Instance : un graphe $G = (S, A)$, deux sommets $s, t \in S$ et un entier k.
   Question : existe-t-il un chemin élémentaire de s à t de longueur $\geq k$?

4. `CHEMIN-`$\geq$`-ARBRE`
   Instance : un arbre $G = (S, A)$.
   Question : existe-t-il un chemin élémentaire de s à t de longueur $\geq k$?

5. `CHEMIN-HAMILTONIEN`
   Instance : un graphe $G = (S, A)$.
   Question : G admet-il un chemin hamiltonien, c'est-à-dire un chemin passant exactement une fois par chaque sommet ?

6. `COUPLAGE-PARFAIT-BIPARTI`
   Instance : un graphe biparti $G = (S, A)$.
   Question : G admet-il un couplage parfait ?

V k-COLOR

Soit $G = (S, A)$ un graphe non orienté. On appelle k-coloration de G une fonction $c : S \longrightarrow \{1, 2, \ldots, k\}$ telle que pour tout arc $(u, v) \in A$, on a $c(u) \neq c(v)$.
Pour $k \in \mathbb{N}^*$, on considère le problème suivant :

k-COLOR

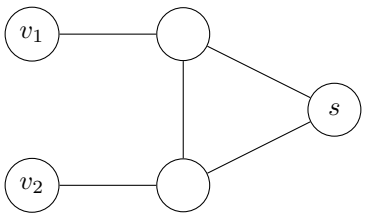
Entrée : un graphe $G = (S, A)$ non orienté
Sortie : G est-il k-colorable ?

1. Montrer que `1-COLOR` et `2-COLOR` appartiennent à P.
2. Montrer que `3-COLOR` appartient à NP.
3. Montrer que `3-COLOR` se réduit polynomialement à `3-SAT`.

Dans la suite, on veut trouver une réduction polynomiale de `3-SAT` à `3-COLOR`.
On considère une formule φ de `3-SAT` de variables $x_1, ..., x_n$ et on veut construire un graphe G qui soit 3-colorable si et seulement si φ est satisfiable.
On ajoute n sommets dans G (encore appelés x_1, ..., x_n par abus de notation) correspondant à x_1, ..., x_n, n sommets correspondant à $\neg x_1$, ..., $\neg x_n$ et 3 sommets V, F, B reliés deux à deux.
Dans un 3-coloriage de G, S et F doivent être de couleurs différentes. Chaque variable x_i sera considérée comme fausse si le sommet correspondant est de la même couleur que F et vraie s'il est de la même couleur que V.

4. Expliquer comment ajouter des arêtes à G pour que chaque variable x_i soit vraie ou fausse (c'est-à-dire coloriée avec la même couleur que F ou la même couleur que V) et de valeur opposée à $\neg x_i$.

On considère un sous-graphe (*gadget*) de la forme suivante à ajouter dans G :



5. Montrer que si v_1 et v_2 sont de la même couleur que F alors la couleur de s est imposée et préciser cette dernière.
6. Montrer que si v_1 ou v_2 est de la même couleur que V alors il existe un coloriage de G où s est de la même couleur que V.
7. Quelle formule logique le gadget ci-dessus permet-il de représenter ?
8. Quel gadget ajouter à G de façon pour représenter une clause $\ell_1 \vee \ell_2 \vee \ell_3$?
9. Montrer que `3-COLOR` est NP-complet.
10. Montrer que k-`COLOR` est NP-complet pour $k \geq 4$.