COMPOSITION D'INFORMATIQUE n°1

Sujet 2 (durée : 4 heures)

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

⋆ ⋆ ⋆

Ce sujet est constitué d'un premier problème sur l'indécidabilité (à faire en 1h20 environ) et d'un deuxième problème sur les graphes (à faire en 2h40 environ) qui sont indépendants.

Problème 1 : théorème de Rice

Dans l'ensemble de ce problème, on manipule des fonctions écrites dans un langage de programmation, par exemple OCaml. On distinguera la notation f, correspondant à la fonction elle-même, ou l'algorithme, et la notation $\langle f \rangle$, désignant le **code source** de la fonction f. Ainsi, si on considère le code :

```
let rec f lst = match lst with
    | []     -> 0
    | _ :: q -> 1 + f q
```

Alors on distingue l'objet `f`, de signature `'a list -> int`, et l'objet ⟨`f`⟩, de signature `string`, correspondant à la chaîne de caractères :

```
"let rec f lst = match lst with | [] -> 0 | _ :: q -> 1 + f q"
```

Pour simplifier l'étude, on suppose que l'ensemble des fonctions manipulées sont de signature `string -> bool`, sauf une fonction `universel : string -> string -> bool` telle que l'appel à `universel` ⟨`f`⟩ `x` simule l'exécution de `f x`. On notera Σ^* l'ensemble des chaînes de caractères.

Si f est une fonction, on note $L(f)$, appelé **langage de** f, l'ensemble des arguments pour lesquels la fonction renvoie `true`, c'est-à-dire :

$$L(f) = \{x \in \Sigma^* \mid f(x) \text{ termine et renvoie } \texttt{true}\}$$

Question 1 Montrer que le problème `Appartient` suivant est indécidable et semi-décidable :

* **Instance** : un code source $\langle f \rangle \in \Sigma^*$ et un argument $x \in \Sigma^*$.
* **Question** : est-ce que $x \in L(f)$?

Question 2 Montrer que le problème `Diagonal` suivant n'est pas semi-décidable :

* **Instance** : un code source $\langle f \rangle \in \Sigma^*$.
* **Question** : est-ce que $\langle f \rangle \notin L(f)$?

Indication : on pourra supposer qu'il est semi-décidable, résolu partiellement par un algorithme $\mathcal{A}$, et s'intéresser à $\langle \mathcal{A} \rangle$ et $L(\mathcal{A})$.

Question 3 Montrer que `Diagonal` $\leqslant_m$ co`Appartient`. Que peut-on en déduire sur les problèmes complémentaires de `Diagonal` et `Appartient` ?

On appelle **propriété des langages de fonctions** tout sous-ensemble de $\mathcal{P}(\Sigma^*)$. Si $P \subseteq \mathcal{P}(\Sigma^*)$, on assimile P et le problème de décision suivant, qu'on notera également P :

* **Instance** : un code source $\langle f \rangle \in \Sigma^*$.

* **Question** : est-ce que $L(f) \in P$?

Question 4 Est-ce que `Diagonal` est une propriété des langages de fonctions ? Justifier.

On appelle **langage** toute sous-ensemble de Σ^*. Si $L \subseteq \Sigma^*$, on assimile L et le problème de décision suivant, noté également L :

* **Instance** : une chaîne $x \in \Sigma^*$.
* **Question** : est-ce que $x \in L$?

Pour la suite, on cherche à montrer le théorème de Rice suivant :

Théorème

Soit P une propriété non triviale des langages semi-décidables. Alors P n'est pas décidable.

Ici, « propriété non triviale des langages semi-décidables » signifie qu'il existe deux langages semi-décidables L_1 et L_2 tels que $L_1 \in P$ et $L_2 \notin P$ (ce n'est ni une propriété vérifiée par aucun langage semi-décidable, ni par tous). On pose P une telle propriété non triviale des langages semi-décidables. Comme P est non triviale, soit $L \in P$ semi-décidable. Comme L est semi-décidable, soit f_L un algorithme qui résout partiellement L.

Question 5 Justifier qu'on peut supposer, sans perte de généralité, que $\emptyset \notin P$. On fera cette hypothèse pour la suite.

Soient $\langle f \rangle \in \Sigma^*$ et $x \in \Sigma^*$. On définit la fonction suivante :

```
let g y =
    universel <f> x && universel <f_L> y
```

Question 6 En considérant la fonction `g`, montrer que `Appartient` $\leqslant_m P$.

Question 7 En déduire le théorème de Rice.

Question 8 Montrer que le problème suivant est indécidable :

* **Instance** : un code source $\langle f \rangle \in \Sigma^*$.
* **Question** : est-ce que $L(f) \neq \emptyset$?

Question 9 On considère le problème suivant :

* **Instance** : un code source $\langle f \rangle \in \Sigma^*$.
* **Question** : est-ce que le code source de f contient au moins 5 boucles `while` ?

a) Ce problème est-il décidable ?
b) Cela rentre-t-il en contradiction avec le théorème de Rice ?

Problème 2 : tas de sable

Les questions de programmation doivent être traitées en langage OCaml. On autorisera toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max`, `incr` ainsi que les opérateurs comme `@`). Sauf précision de l'énoncé, l'utilisation d'autres modules sera interdite.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n, m))$ où n et m sont les tailles des arguments de la fonction, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.