Feuille d'exercices n°5 - Décidabilité

Notions abordées

- sérialisation
- réduction entre problèmes
- variantes du problème de l'arrêt
- exemples de problèmes décidables, exemples de problèmes indécidables

Attention : Par souci de lisibilité, dans tout le TD, lorsque `m` est une variable de type `str` d'un programme OCaml on note `"blabla {{m}} exemple"` la chaîne `"blabla " ^ m ^ "exemple"`.
De plus on pourra utiliser les fonctions `execute` et `serialise_couple`, `deserialise_couple` introduites en cours.

Exercice 1 : Sérialisation de types énumérés

Q. 1 Si un type `t` admet une sérialisation calculable, montrer que le type `t list` des listes d'éléments de type `t` admet une sérialisation calculable.

Q. 2 Soit $m \in \mathbb{N}$, une suite $(n_i)_{i\in[\![1,m]\!]}$ d'entiers naturels, des types $(\mathtt{t}_{i,j})_{i\in[\![1,m]\!],j\in[\![1,n_i]\!]}$, et des fonctions de sérialisations calculables $\varphi_{i,j} : \mathtt{t}_{i,j} \to \Sigma_{\mathsf{bp}}^\star$ pour chacun de ces types. On définit alors un type :

```
type enumeration =
  | C_1 of t_{1,1} * t_{1,2} * ... * t_{1,n_1}
  | C_2 of t_{2,1} * t_{2,2} * ... * t_{2,n_2}
  | ...
  | C_m of t_{m,1} * t_{m,2} * ... * t_{m,n_m}
```

Donner une fonction de sérialisation calculable du type `enumeration`.

Exercice 2 : Stabilité de la classe des langages décidables

Q. 1 Montrer que tout langage fini est décidable.

Q. 2 Montrer que l'ensemble des langages décidables est stable par concaténation.

Q. 3 La question précédente assure en particulier que si L est un langage décidable, alors $L\cdot L$ est aussi décidable. La réciproque est-elle vraie ?

Q. 4 Montrer que l'ensemble des langages décidables n'est stable ni par union dénombrable ni par intersection dénombrable.

Q. 5 Soit un langage $L \subseteq A \times B$ décidable. Le langage $\pi_A(L) = \{a \in A \mid \exists b \in B, (a,b) \in L\}$ est-il nécessairement décidable ?

Exercice 3 : Non monotonie du caractère décidable des langages

Q. 1 Donner trois langages A, B, C sur le même alphabet Σ tels que $A \subseteq B \subseteq C$, A et C sont décidables, mais B est indécidable.

Q. 2 Donner trois langages A, B, C sur le même alphabet Σ tels que tels que $A \subseteq B \subseteq C$, A et C sont indécidables, mais B est décidable.

Exercice 4 : Réductions

Montrer que les problèmes suivants sont indécidables.

1. ArrêtUniv
   Entrée : La sérialisation d'une machine $\mathcal{M}$
   Sortie : La machine $\mathcal{M}$ termine-t-elle sur toutes ses entrées ?
2. ArrêtSimult
   Entrée : La sérialisation d'un couple de machines $(\mathcal{M}, \mathcal{N})$
   Sortie : Les machines $\mathcal{M}$ et $\mathcal{N}$ terminent-t-elles sur les mêmes entrées ?
3. Régulier
   Entrée : La sérialisation d'une machine $\mathcal{M}$
   Sortie : Le langage $\mathcal{L}(\mathcal{M})$ est-il régulier ?
4. $w \in \Sigma^\star$, Arrêt_w
   Entrée : La sérialisation d'une machine $\mathcal{M}$
   Sortie : La machine $\mathcal{M}$ s'arrête-t-elle sur w ?
5. ArrêtExiste
   Entrée : La sérialisation d'une machine $\mathcal{M}$
   Sortie : La machine $\mathcal{M}$ s'arrête-t-elle sur au moins une entrée ?

Exercice 5 : Quelques problèmes décidables

Q. 1 Montrer que pour toute fonction $f : \mathbb{R} \to \mathbb{R}$ le problème suivant est décidable.

Zero_f :
Entrée : Rien
Sortie : Existe-t-il $x \in \mathbb{R}$ tel que $f(x) = 0$?

Q. 2 Montrer que pour toute machine $\mathcal{M}$ et pour tout mot $w \in \Sigma^\star$, le problème suivant est décidable.

$\text{Arrêt}_{\mathcal{M},w}$:
Entrée : Rien
Sortie : L'exécution de $\mathcal{M}$ sur w s'arrête-t-elle ?

Q. 3 Le problème suivant est-il décidable ?
Entrée : La sérialisation d'une machine $\mathcal{M}$
Sortie : Existe-t-il une machine $\mathcal{N}$ telle que $\mathcal{N} \neq \mathcal{M}$ et $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$?

Exercice 6 : Amélioration de code

Dans cet exercice on s'intéresse au problème d'améliorer automatiquement le code que vous produisez en séances de TP.

Fonction morte. On appelle **fonction morte** une fonction d'un programme OCaml qui n'est jamais exécutée, et ce quelle que soit l'entrée du programme. On s'intéresse au problème de détecter

automatiquement de telles fonctions afin de les supprimer. Ce problème étant difficile à formaliser, on s'intéresse plutôt au problème suivant qui est une variante plus simple.

$$\textsc{CodeMort} : \begin{cases} \textbf{Entrée :} & \text{La sérialisation d'un programme OCaml } M \\ \textbf{Sortie :} & \text{Existe-t-il une définition de fonction locale } d = \texttt{let f ... = ... in} \\ & \text{telle que } \langle M \rangle = u \cdot d \cdot v \text{ et } u \cdot v \text{ est la sérialisation d'un programme} \\ & \text{OCaml se comportant comme } M \text{ sur toutes ses entrées ?} \end{cases}$$

Q. 1 Donner un exemple de fonction morte.

Q. 2 Montrer que ce problème est indécidable.

Propagation des constantes. Étant donné un programme OCaml, et un ensemble de noms de variables $\mathcal{V}$, on appelle **variable constante de** $\mathcal{V}$ une variable d'identifiant dans $\mathcal{V}$ de type `'a ref` dont la valeur n'est jamais modifiée par l'exécution du programme, et ce quelle que soit l'entrée du programme. Détecter les variables constantes dans un programme permet par exemple de remplacer l'utilisation de variables par les valeurs des constantes qu'elles contiennent.

Q. 3 Donner un exemple de programme contenant une variable constante, et de ce même programme dans lequel la constante a été remplacée par sa valeur.

On ne demande pas ici de formaliser le problème.

Q. 4 Montrer que ce problème est indécidable, par réduction depuis CodeMort.

Exercice 7 : Théorème de Rice

On s'intéresse dans cet exercice aux prédicats sur les langages, et aux problèmes associés de reconnaître les machines dont le langage vérifie ce prédicat.
Intuitivement un **prédicat** sur les langages exprime une propriété, qui peut être vérifiée ou non (être vide, fini, infini, stable par telle ou telle opération, contenir un autre langage...). Formellement c'est donc un ensemble de langages (ceux qui vérifient le prédicat). En effet si $L \subseteq \Sigma^\star$ est un langage sur Σ, on dit que L **vérifie le prédicat** $\mathcal{Q} \subseteq \mathcal{P}(\Sigma^\star)$ si et seulement si $L \in \mathcal{Q}$.

Étant donné une machine $\mathcal{M}$, son langage, noté $\mathcal{L}(\mathcal{M})$ est défini en cours. Si le langage d'une machine $\mathcal{M}$ vérifie $\mathcal{Q}$, *i.e.* si $\mathcal{L}(\mathcal{M}) \in \mathcal{Q}$, on dira par extension que $\mathcal{M}$ vérifie $\mathcal{Q}$. On note que le fait que $\mathcal{M}$ vérifie $\mathcal{Q}$ ne dépend pas de la machine $\mathcal{M}$ elle-même, mais seulement de son langage. Autrement dit, si $\mathcal{M}$ et $\mathcal{N}$ sont telles que $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$, alors : $\mathcal{M}$ vérifie $\mathcal{Q} \Rightarrow \mathcal{N}$ vérifie $\mathcal{Q}$.

Pour un prédicat $\mathcal{Q} \subseteq \mathcal{P}(\Sigma^\star)$, on s'intéresse au problème de décision suivant.

$$\textsc{Vérifie}_{\mathcal{Q}} : \begin{cases} \textbf{Entrée} & : & \text{La sérialisation d'une machine } \mathcal{M} \\ \textbf{Sortie} & : & \mathcal{M} \text{ vérifie-t-elle } \mathcal{Q} \text{ ?} \end{cases}$$

De plus on dit d'un prédicat $\mathcal{Q} \subseteq \mathcal{P}(\Sigma^\star)$ qu'il est **trivial sur les langages des machines** si et seulement si tous les langages de machines le vérifient ou bien aucun langage de machine ne le vérifie.
Le théorème de Rice assure que **si $\mathcal{Q}$ est un prédicat non trivial sur les langages des machines, alors Vérifie$_{\mathcal{Q}}$ est indécidable**. Autrement dit, le problème de savoir, à partir du code d'une machine, si le langage des mots qu'elle accepte vérifie une propriété est indécidable, sauf si la propriété est toujours vraie ou au contraire toujours fausse.

Q. 1 Démontrer le théorème de Rice. ♣

Q. 2 Dans chacun des cas suivants dire si le problème est décidable ou non.
Si le langage est décidable justifier laquelle des hypothèses du théorème de Rice n'est pas vérifiée.

1. **Entrée** : La sérialisation d'une machine $\mathcal{M}$
   Sortie : Le langage $\mathcal{L}(\mathcal{M})$ est-il $\Sigma^\star$?
2. **Entrée** : La sérialisation d'une machine $\mathcal{M}$
   Sortie : La machine $\mathcal{M}$ contient-elle une boucle `while` ?
3. **Entrée** : La sérialisation d'une machine $\mathcal{M}$
   Sortie : Le langage $\mathcal{L}(\mathcal{M})$ contient-il un nombre pair de mots ?
4. **Entrée** : La sérialisation d'une machine $\mathcal{M}$
   Sortie : Existe-t-il une machine $\mathcal{N}$ telle que $\mathcal{N} \neq \mathcal{M}$ et $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$?

Exercice 8 : Un changement de modèle de calcul

Afin de comprendre l'impact du modèle de calcul dans la notion de calculabilité, on définit ici un modèle de calcul alternatif à celui présenté en cours.

Super-machine. Une **super-machine** est une fonction OCaml `f` de type `int -> string -> bool`. L'exécution d'une telle super-machine $\mathcal{M}$ sur une entrée $w \in \Sigma^*$ est définie comme produisant un comportement d'erreur s'il existe $n_e \in \mathbb{N}$ tel que l'appel OCaml `(f` n_e w`)` conduit à un comportement d'erreur, *i.e.* déclenche une erreur ou boucle indéfiniment. Lorsque l'exécution d'une super-machine $\mathcal{M}$ sur une entrée $w \in \Sigma^*$ ne conduit pas à un comportement d'erreur, on définit la valeur de l'exécution de $\mathcal{M}$ sur w comme étant : V si et seulement s'il existe $n_V \in \mathbb{N}$ tel que l'appel OCaml `(f` n_V w`)` vaut `true` et F sinon. Afin d'éviter les ambiguïtés dans la suite, on parlera de la **super-exécution** d'une super-machine.

Q. 1 Démontrer que le problème de l'arrêt **d'une machine du cours** est décidable par une super-machine. Comprendre : il existe une super-machine, dont la super-exécution termine sans erreur sur toute entrée, et telle que, lorsqu'on la super-exécute sur la sérialisation d'un couple $(\mathcal{M}, w)$ où $\mathcal{M}$ est une machine du cours, et $w \in \Sigma^*$, elle retourne V si et seulement si la machine $\mathcal{M}$ s'arrête lors de son exécution sur l'entrée w.

Puisqu'une super-machine est une fonction OCaml, elle peut être sérialisée au moyen d'une chaîne de caractères. On se pose alors la question : une super-machine peut-elle décider de l'arrêt d'une super-machine ?

Q. 2 Démontrer que le problème de l'arrêt d'une super-machine est indécidable par une super-machine.

♣. Si cette question est trop difficile, par manque d'exemple de ce que pourrait être une telle propriété $\mathcal{Q}$, on pourra regarder la question 2.