TD 08

Décidabilité

MPI/MPI*, lycée Faidherbe

On rappelle que le problème de l'arrêt, ARRÊT, est indécidable

Instance : un programme f et un argument e.

Question : l'exécution de f sur e termine-t-il ?

I Réductions du problème de l'arrêt

Exercice 1 **Solution page 4**

Prouver que le problème ARRÊT (VIDE)

Instance : une fonction f.

Question : est-ce que l'exécution de f sans argument termine en temps fini ?

est indécidable

Exercice 2 **Solution page 4**

Prouver que le problème ARRÊT 2

Instance : une fonction f et deux arguments x et y.

Question : est-ce que les calculs de $f(x)$ et $f(y)$ terminent en temps fini ?

est indécidable

Exercice 3 **Solution page 4**

Prouver que le problème COARRÊT

Instance : une fonction f et un argument x.

Question : le calcul de $f(x)$ est-il infini ?

est indécidable

Exercice 4 **Solution page 4**

Prouver que le problème ZÉRO

Instance : une fonction f et un argument x.

Question : l'exécution de f sur x renvoie-t-elle 0 ?

est indécidable

Exercice 5 **Solution page 4**

Prouver que le problème ARRÊT (EXISTE)

Instance : une fonction f.

Question : existe-t-il une entrée x telle que l'exécution de $f(x)$ termine ?

est indécidable

Exercice 6 **Solution page 4**

Prouver que le problème ARRÊT (TOUT)

Instance : une fonction f.

Question : l'exécution de $f(x)$ termine-t-elle pour toute entrée x ?

est indécidable

II Semi-décidabilité

Définition

Un problème de décision est *semi-décidable* s'il existe un algorithme qui termine en renvoyant `true` sur toute instance positive et ne termine pas ou renvoie `false` sur toute instance négative.

Si on n'a pas de réponse de l'algorithme après un certain temps, on ne sait pas si cela signifie qu'il ne termine pas ou qu'une réponse positive va venir.

Exercice 7 Solution page 5
Prouver que tout problème décidable est semi-décidable

Exercice 8 Solution page 5
Prouver que ARRÊT est semi-décidable

Définition

Si P est un problème de décision, son problème *complémentaire* est coP ayant les mêmes instances mais dont la question est la négation de la question de P.

Dans l'exercice 3, on a le problème complémentaire de ARRÊT.

Exercice 9 Solution page 5
Prouver que si coP $\leqslant$ P et P $\leqslant$ coP.

Exercice 10 Solution page 5
Prouver que si P et coP sont semi-décidables alors P est décidable.

Exercice 11 Solution page 5
Prouver que coARRÊT n'est pas semi-décidable.

De manière générale, si P est indécidable mais semi-décidable alors coP n'est pas semi-décidable (et n'est pas décidable non plus). Cela montre que, si on a $A \leqslant B$ avec A semi-décidable, on n'a pas forcément B semi-décidable ; prendre A = coARRÊT et B = ARRÊT. Cependant

Exercice 12 Solution page 5
Prouver que si A se réduit à B par transformation des instances ($A \leqslant_m B$) et si B est semi-décidable alors A est semi-décidable.

Exercice 13 Solution page 5
Prouver que le problème ARRÊT (TOUT) n'est pas semi-décidable (voir l'exercice 6).

III Théorème de Rice

Définition

Deux programmes sont *équivalents* s'ils sont définies sur les mêmes entrées et si s'ils terminent sur les mêmes valeurs en renvoyant le même résultat.
Un problème de décision portant sur des fonctions est *sémantique* si sa réponse est la même pour deux fonctions équivalentes.

Soit P un problème de décision sémantique décidable, décidé par une fonction `phi_P` (Φ_P).
On considère la fonction u, qui ne termine sur aucune entrée

```
let u x =
   while true do () done
```

Pour toute fonction v on définit

```
let essai f x =
   let w y =
      let _ = f x in
      v y
   in phi_P w <> phi_P u
```

Exercice 14 **Solution page 6**
Déterminer une fonction équivalente à w selon que $f(x)$ termine ou non.
En déduire que `essai f x` renvoie `false` si $f(x)$ ne termine pas.

Exercice 15 **Solution page 6**
En déduire que Φ_P est constante.

On a donc prouvé la contraposée du

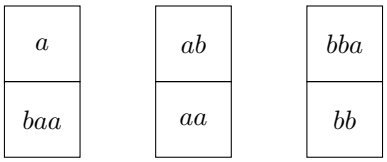
Théorème de Rice

Tout problème de décision sémantique non constant est indécidable.

IV Correspondance de Post

Un exemple

On se donne 3 modèle de dominos et on essaye, en alignant des dominos de ce type, d'obtenir le même mot en haut et en bas.
$(a, baa), (ab, aa), (bba, bb)$



Exercice 16 **Solution page 6**
Trouver un alignement de 4 dés qui donne une solution.

De manière plus formelle le problème (de décision) de la correspondance de Post (PCP) est

Instance : une suite finie de couples de mots $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$.

Question : existe-t-il un entier p et une fonction k de $\{1, 2, \ldots, p\}$ vers $\{1, 2, \ldots, n\}$ tels que $u_{k(1)}u_{k(2)} \cdots u_{k(p)} = v_{k(1)}v_{k(2)} \cdots v_{k(p)}$?

On admet que PCP est indécidable.

Si $(u_1, v_1), ..., (u_n, v_n)$ est une instance de PCP avec $u_i, v_i \in \Sigma^*$, on introduit un alphabet $A = \{a_1, ..., a_n\}$ dont les lettres n'appartiennent pas à Σ.
On note L_u le langage sur $\Sigma \cup A$ $L_u = \{u_{i_1}...u_{i_k}a_{i_k}...a_{i_1}$; $k > 0$ et $1 \leqslant i_r \leqslant n$ pour tout $r\}$.

Exercice 17 **Solution page 6**
Prouver que L_u est algébrique.

Exercice 18 **Solution page 6**
Montrer l'indécidabilité du problème INTERSECTION_VIDE :

Instance : deux grammaires algébriques G et G'.

Question : l'intersection de $L(G)$ et $L(G')$ est-elle vide ?

Exercice 19 **Solution page 6**
Montrer l'indécidabilité du problème AMBIGUË :

Instance : une grammaire algébrique G.

Question : G est-elle ambiguë ?

Solutions

Exercice 1
S'il existait une fonction `arret_vide : (unit -> 'a) -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
    let g () = f x in
    arret_vide g;;
```

C'est une transformation des instances.

Exercice 2
S'il existait une fonction `arret2 : (string -> string) -> string -> string -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
    arret2 f x x;;
```

C'est une transformation des instances.

Exercice 3
S'il existait une fonction `coArret : (string -> string) -> string -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
    not (coArret f x);;
```

Exercice 4
S'il existait une fonction `coArret : (string -> string) -> string -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
       let g x = let _ = f x in 0
       in zero g x
```

C'est une transformation des instances.

Exercice 5
S'il existait une fonction `arret_existe : (string -> string) -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
       let g y = f x in
       arret_existe
```

C'est une transformation des instances.

Exercice 6
S'il existait une fonction `arret_tout : (string -> string) -> bool` résolvant le problème alors on pourrait construire une fonction résolvant le problème de l'arrêt

```
let arret f x =
       let g y = f x in
       arret_tout g
```

C'est une transformation des instances.

Exercice 7
Une fonction qui décide le problème convient pour la semi-décidabilité.

Exercice 8
Une fonction possible est

```
let arret f x =
   let _ = f x in
   true
```

Si f termine sur l'entrée x alors `arret` renvoie `true` sinon `arret` ne termine pas.

Exercice 9
Si on a une fonction `decid_P` qui décide P alors la fonction

```
let decid_coP x =
   lnot (decid_P x)
```

décide coP.
Comme on a cocoP = P la deuxième partie en découle.

Exercice 10
On verra dans l'année la possibilité d'exécuter deux programmes en parallèle ; si on exécute simultanément un programme qui répond à la semi-décidabilité de P et un programme qui répond à la semi-décidabilité de coP, l'un des deux va s'arrêter et on renvoie la réponse adapter (en interrompant le second).

Exercice 11
Si coARRÊT était semi-décidable ARRÊT serait décidable.

Exercice 12
Si f_B est une fonction qui permet la semi-décidabilité de B et si Φ est une fonction calculable qui transforme les instances x pour A en instances de B en conservant la positivité, $f_A = f_B \circ \Phi$ permet la semi-décidabilité de A.

Exercice 13
On va réduire coARRÊT à ARRÊT (tout) par transformation des instances.
Soit (f, x) une instance de coARRÊT. On définit g une fonction qui prend un argument entier n et qui calcule $f(x)$ en s'arrêtant avant les n premières opérations :

- si le calcul était terminé, boucler à l'infini ;
- sinon renvoyer 0.

Si (f, x) est une instance positive de coARRÊT, alors le calcule de (x) ne termine jamais donc, pour tout n $g(n)$ termine et g est une instance positive de ARRÊT (tout).
Si (f, x) est une instance négative de coARRÊT alors il existe un entier N tel que $f(x)$ termine avant N opération donc $g(n)$ ne termine pas pour $n \geqslant N$: g est une instance négative de ARRÊT (tout).

On a bien une réduction many-to-one et on peut appliquer l'exercice précédent.

Exercice 14
Si $f(x)$ termine, w est équivalente à v.
Si $f(x)$ ne termine pas, w est équivalente à u donc le résultat de `essai f x` est `false`

Exercice 15
Si $f(x)$ termine, le résultat de `essai f x` ne peut être `true` car sinon on aurait une fonction décidant le problème de l'arrêt. Ainsi `phi_P w <> phi_P u` doit renvoyer `false` d'où $\Phi_P(w) = \Phi_P(u)$
Or $\Phi_P(w) = \Phi_P(v)$ car v et w sont équivalentes dans ce cas, donc on doit avoir $\Phi_P(v) = \Phi_P(u)$ pour tout v : Φ_P est constante.

Exercice 16

bba	ab	bba	a
bb	aa	bb	baa

Exercice 17
L_u est engendré par $S \to u_1 S a_1 | u_2 S a_2 | \cdots | u_n S a_n | u_1 a_| u_2 a_2 | \cdots | u_n a_n$.

Exercice 18
On réduit PCP par transformation des instances. À toute instance de PCP on associe les langages algébrique $L(u)$ et $L(v)$. L'instance est positive si et seulement si $L(u) \cap L(V)$ est non vide.
Ainsi PCP $\leqslant$ coINTERSECTION_VIDE donc coINTERSECTION_VIDE est indécidable puis INTERSECTION_VIDE est indécidable.

Exercice 19
À partir d'une instance de PCP, on définit $L(U) \cup L(V)$ est algébrique.
Ce langage est engendré par la grammaire $S \to S_u | S_v$, $S_u \to u_1 S_u a_1 | \cdots$, $S_v \to v_1 S_v a_1 | \cdots$
L'instance de PCP est positive si et seulement si la grammaire est ambiguë donc PCP $\leqslant$ AMBIGUË.