

TD5. Décidabilité

Exercice 1. Problèmes d'optimisations

1. Sac à Dos $\left\{ \begin{array}{l} \text{Entrée :} \quad \text{Une liste d'objets } (p_i, v_i)_{1 \leq i \leq n} \text{ et une capacité } C \in \mathbb{N}^*, K \in \mathbb{N} \\ \text{Sortie :} \quad \text{Existe-t-il } I \subseteq \{1, \dots, n\} \text{ tq } \sum_{i \in I} p_i \leq C \text{ et } \sum_{i \in I} v_i \geq K ? \end{array} \right.$
2. Couverture Sommets $\left\{ \begin{array}{l} \text{Entrée :} \quad G = (S, A) \text{ un graphe non orienté, } K \in \mathbb{N} \\ \text{Sortie :} \quad \text{Existe-t-il } C \subseteq S \text{ tq } \forall (u, v) \in A, (u \in C) \text{ ou } (v \in C) \text{ et } |C| \leq K ? \end{array} \right.$

Exercice 2. Stabilité de la classe des langages décidables

On dit que L est un **langage décidable** s'il existe une machine m telle que $L(m) = L$.

Soit L_1 et L_2 deux langages décidables.

1. Soit L_1 et L_2 décidables. Alors il existe m_1 tq $L_1 = L(m_1)$ et m_2 tq $L_2 = L(m_2)$.

On rappelle qu'il existe une fonction `eval` : `int` -> `string` -> `string` -> `bool` telle que `eval k <m> e` termine et renvoie `true` si `m e` termine en moins de `k` étapes, et `false` sinon.

On définit alors la fonction `m` suivante :

```

1 let m e =
2   let i = ref 1 in
3   try
4     while true do
5       if eval !i m1 e || (eval !i m2 e) then raise Exit ;
6       i := !i + 1
7     done;
8     false
9   with |Exit -> true

```

Montrons que $L(m) = L_1 \cup L_2$.

$\subseteq$: Soit $e \in L(m)$. Alors `m e` termine, donc la boucle a été interrompue, donc la condition du `if` est évaluée à `true`, donc il existe `i` tel que `eval i m1 e` termine ou `eval i m2 e` termine, donc `m1 e` termine ou `m2 e` termine, donc $e \in L(m_1) \cup L(m_2)$.

$\supseteq$: Soit $e \in L(m_1) \cup L(m_2)$. On suppose sans perte de généralité $e \in L(m_1)$. Alors `m1 e` termine en un nombre fini d'opérations, que l'on note `k`. Donc `eval k m1 e` renvoie `true`, et au `k` ième tour de boucle, `m e` renvoie `true`. Ainsi, $e \in L(m)$.

2. Montrer que $L_1.L_2$ est décidable.

```

1 let m e =
2   let i = ref 1 in
3   let m = String.length e in
4   try
5     while true do
6       for j = 0 to m do
7         let e1 = String.sub e 0 j in
8         let e2 = String.sub e j (m-j) in
9         if eval !i m1 e1 && (eval !i m2 e2) then raise Exit ;
10      done;
11      i := !i + 1
12    done;
13    false
14  with |Exit -> true

```

3. Raisonnons par l'absurde et supposons la classe des langages décidables stable par union dénombrable.

Pour tout $k \in \mathbb{N}$, le langage $L_k = \{ \langle m, e \rangle \mid m \text{ e termine en moins de } k \text{ étapes} \}$ est décidable ($L_k = L(\text{eval } k)$). Donc $L = \bigcup_{k \in \mathbb{N}} L_k = \{ \langle m, e \rangle \mid m \text{ e termine} \}$ est décidable. La machine associée à L décide le problème de l'arrêt, ce qui est absurde.

Exercice 4. Décidables ? Indécidables ?

Montrons que $\text{Arrêt} \leq \Pi_3$. Soit $\langle m_0 \rangle, e_0$ une instance de Arrêt. On construit $\langle m \rangle$ une instance de Π_3 définie par

```

1 let m e =
2   if e = 1 then true
3   else
4     let _ = m0 e0 in
5     e = 0

```

La réduction est décidable. Montrons que $m \in \Pi_2^+$ ssi $(\langle m_0 \rangle, e_0) \in \text{Arrêt}^+$.

Si $m \in \Pi_2^+$, alors $L(m)$ contient un nombre pair de mots. Supposons par l'absurde que $m_0 e_0$ ne termine pas. Alors $L(m) = \{1\}$, donc $m \notin \Pi_2^+$.

Si $(\langle m_0 \rangle, e_0) \in \text{Arrêt}^+$, alors $L(m) = \{1, 0\}$, donc $m \in \Pi_2^+$.

Ainsi, $\text{Arrêt} \leq \Pi_3$. Comme Arrêt est indécidable, Π_3 aussi.