IFT-2002

Informatique Théorique

H14 - cours 12

Julien Marcil - julien.marcil@ift.ulaval.ca

Preuve qu'un langage L est non régulier

Pour prouver qu'un langage est non régulier, on fait une preuve par contradiction.

- On suppose que L est régulier.
- Donc il existe p la longueur de pompage de L.
- On *choisi* un mot $w \in L$, avec $|w| \geq p$ (qu'il ne sera pas possible de pomper).
- On montre qu'il n'existe pas de valeurs x, y, z parce qu'en pompant y, on génère des mots xy^iz qui ne sont pas dans L.

Exemple

Soit $\Sigma = \{0, 1\}$. Prouver que le langage
$L = \{w \mid w \text{ contient le même nombre de } 0 \text{ et de } 1\}$ n'est pas régulier.

Langage hors contexte

Définition: Un langage est dit **hors contexte** (ou *non contextuelle*) s'il existe une grammaire hors contexte qui le génère.

Lemme de pompage

Si L est un langage hors contexte, alors il existe un entier $p \geq 1$ (appelé longueur de pompage) tel que pour tout mot $w \in L$ avec $|w| \geq p$, il existe des mots u, v, x, y, z tels que $w = uvxyz$ et

1. $|vxy| \leq p$
2. $|vy| > 0$
3. pour tout entier $i \geq 0$ on a $uv^i xy^i z \in L$

Preuve qu'un langage L est non hors contexte

Pour prouver qu'un langage est non hors contexte, on fait une preuve par contradiction.

- On suppose que L est hors contexte.
- Donc il existe p la longueur de pompage de L.
- On *choisi* un mot $w \in L$, avec $|w| \geq p$ (qu'il ne sera pas possible de pomper).
- On montre qu'il n'existe pas de valeurs u, v, x, y, z parce qu'en pompant v et y, on génère des mots $uv^i xy^i z$ qui ne sont pas dans L.

Exemple

Soit $\Sigma = \{a, b, c\}$. Prouver que le langage $L = \{w \mid w$ contient le même nombre de a,b et c$\}$ n'est pas régulier.

Arrêt d'une machine de Turing

Une machine de turing peut:

- se rendre à un état final et arrêter
- boucler indéfiniment

Turing-acceptable

Définition: Un langage L est dit **Turing-acceptable** (ou *récursivement énumérable*)s'il existe une machine de Turing qui accepte les mots de L.

Étant donnée une entrée w

- Si $w \in L$ alors M s'arrête et accepte w
- Si $w \notin L$ alors M s'arrête et rejette w ou ne s'arrête pas

Turing-décidable

Définition: Un langage L est dit **Turing-décidable** (ou *décidable*)s'il existe une machine de Turing M qui accepte les mots de L et rejette les autres mots.

Étant donnée une entrée w

- Si $w \in L$ alors M s'arrête et accepte w
- Si $w \notin L$ alors M s'arrête et rejette w

Théorème

Soit un le langage L. Si L est *Turing-décidable*, alors $\overline{L}$ est *Turing-décidable*.

Théorème

Soit un le langage L. Si L et $\overline{L}$ sont *Turing-acceptable*, alors L est *Turing-décidable*.

La thèse de Church-Turing

Les règles formelles de calcul (machines de Turing, le λ-calcul, les fonctions récursives...) *formalisent* correctement la notion d'algorithme.

Turing-complet

Un langage informatique est dit **Turing-complet** s'il permet de représenter toutes les fonctions calculables au sens de Turing et Church.

Dans un tel langage, il est possible de programmer n'importe quelle machine de Turing, mais également tout ce que l'on peut programmer dans une machine de Turing.

Réduction

Un language L se réduit au language K, noté $L \leq_m K$ si il existe $f : \Sigma^* \to \Sigma^*$ une fonction calculable tel que

$$\forall_{w \in \Sigma^*} \quad w \in L \Leftrightarrow f(w) \in K$$

Théorème

Si $A \leq_m B$ et B est décidable, alors A est décidable.

Corrolaire

Si $A \leq_m B$ et A n'est pas décidable, alors B n'est pas décidable.

E_{AFD}

$$E_{\text{AFD}} = \{\langle B \rangle \mid B \text{ est un AFD et } L(B) = \emptyset\}$$

Théorème

E_{AFD} est un langage décidable.

EQ_{AFD}

$$EQ_{\text{AFD}} = \{\langle A, B\rangle \mid A \text{ et } B \text{ sont des AFD et } L(A) = L(B)\}$$

Théorème

EQ_{AFD} est un langage décidable.

Théorie des ensembles

Soit C un AFD tel que

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

$$L(A) = L(B) \Leftrightarrow L(C) = \emptyset$$

A_{MT}

$$A_{\text{MT}} = \{\langle M, w\rangle \mid M \text{ est une MT qui accepte } w\}$$

MT : machine de Turing

Théorème

A_{TM} est un langage *Turing-acceptable*.

Théorème

A_{TM} n'est pas un langage décidable.

Corrolaire

$\overline{A_{\mathrm{TM}}}$ n'est pas un langage *Turing-acceptable*.

$HALT_{\text{MT}}$

$$HALT_{\text{MT}} = \{\langle M, w\rangle \mid M \text{ est une MT et } M \text{ s'arrête sur } w\}$$

Théorème

$HALT_{\mathrm{MT}}$ n'est pas un langage décidable.

Addition

$\texttt{PLUS}(r_1, r_2) = r_1 + r_2$

$r_0 \leftarrow r_1$
répéter r_2 fois [
    $\texttt{inc}(r_0)$
]

Multiplication

$\texttt{MULT}(r_1, r_2) = r_1 \times r_2$

```
répéter r1 fois [
   répéter r2 fois [
      inc(r0)
   ]
]
```

Exponentiation

$\texttt{EXP}(r_1, r_2) = r_1^{r_2}$

$r_0 \leftarrow 1$
répéter r_2 fois [
    $r_0 \leftarrow \texttt{MULT}(r_0, r_1)$
]

La classe P

La classe de complexité **P** est définie comme

$$\bigcup_{k \geq 0} \mathsf{TIME}(n^k)$$

La classe P correspond aux langages décidables en pratique en un temps raisonnable.

Vérificateur

Définition: Un **vérificateur polynômial** pour un langage L est une machine de Turing V tel que pour tout $w \in \Sigma^*$:

- si $w \in L$ alors il existe un mot c tel que $\langle w, c \rangle \in L(V)$
- si $w \notin L$ alors pour tout c on a $\langle w, c \rangle \notin L(V)$

le temps de calcul de V sur $\langle w, c \rangle$ est polynômial en la taille de w.

Un c tel que $\langle w, c \rangle \in L(V)$ est appelé *certificat* ou *preuve* ou *temoin* de l'appartenance de w au langage L.

La classe NP

La classe de complexité **NP** est l'ensemble des langages qui possèdent un vérificateur polynômial.

P vs NP

- P : les langages *décidables* efficacement.
- NP : les langages *vérifiables* efficacement.

Réduction polynômiales

Un language L se réduit au language K, noté $L \leq_p K$ si il existe $f : \Sigma^* \to \Sigma^*$ une fonction calculable en temps polynômiales tel que

$$\forall_{w \in \Sigma^*} \quad w \in L \Leftrightarrow f(w) \in K$$