La non-calculabilité

Maria Malek

19 décembre 2010

- Le concept de problème a été formalisé par le langage.
- La notion de programme a été définie en termes de langages décidé par une machine de Turing.
- Certains problèmes sont décidés par une machine de Turing
- D'autres ne sont pas décidés par une machine de Turing.

Les problèmes & les classes de décidabilité - 1

- Définition : La classe de décidabilité R est l'ensemble de langages décidables par une machine de Turing
  - L est décidable par un langage de Turing : $L \in R$.
  - Autrement dit le problème associé est soluble par une procédure effective (un programme).
  - L est la classe des langages : *décidables* , *calculables*, *solubles algorithmiquement*, *récursifs*.
- Définition : La classe de décidabilité RE est l'ensemble de langages acceptés par une machine de Turing.

Les problèmes & les classes de décidabilité - 2

- Définition I : La classe de décidabilité R est l'ensemble de langages décidables par une machine de Turing
- Définition II : La classe de décidabilité RE est l'ensemble de langages acceptés par une machine de Turing.
  - L est accepté par un langage de Turing : $L \in RE$.
  - Autrement dit, il existe un programme qui donne une réponse positive pour les mots du langage et donne une réponse négative ou boucle pour les mots qui ne sont pas dans le langage.
  - Les langages proches de la décidabilité.
  - LR est la classe des langages : *partiellement décidables* , *partiellement calculables*, *partiellement solubles algorithmiquement*, *récursivement énumérables*.
- Lemme 1 : La classe R est contenue dans la classe RE ($R \subseteq RE$)

Un premier langage indécidable - 1

- La diagonalisation : Les mots finis w_j ainsi que les machines de Turing M_i sont dénombrables. On peut donc construire le tableau infini suivant :
  - $A[M_i, w_j] = O(oui)$ si la machine M_i accepte le mot w_j.
  - $A[M_i, w_j] = N(Non)$ si la machine M_i n'accepte pas le mot w_j (le rejette ou boucle).
- Nous définissons le langage L_0 ainsi

$$L_0 = \{w \mid w = w_i \wedge A[M_i, w_i] = N\}$$

- Théorème 1 : Le langage L_0 n'est pas dans la classe RE.

Un premier langage indécidable - 2

A	w_0	w_1	w_2	...	w_j	...
M_0	O	N	N	...	O	...
M_1	N	N	O	...	O	...
M_2	O	O	N	...	N	...
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
M_j	N	N	O	...	O	...
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	

Un premier langage indécidable - 3

- Théorème 1 : Le langage L_0 n'est pas dans la classe RE.
- Démonstration
  - Si ce langage est dans la classe RE alors il existe une machine de Turing qui l'accepte, autrement dit il existe k tel que M_k accepte L_0, Deux cas de figures contradictoires se présentent :
    1. $w_k \in L_0$ mais $A[M_k, w_k] = O$!
    2. $w_k \notin L_0$ mais $A[M_k, w_k] = N$!

Lemme : Le complément d'un langage récursif est récursif

- Lemme 2 : Le complément d'un langage de la classe R est dans R.
- Démonstration
  - Si L est un langage de la classe R. Il est décidé par une machine de Turing M.
  - Il est simple de construire la machine M' à partir de M qui décide le complément de L : $\bar{L}$.
  - La machine M' est construite de façon à ce que les réponses *(oui, non)* soient inversées.

Lemme : Le complément d'un langage récursivement énumérable ?

- Lemme 3 : Si les deux langages L et $\bar{L}$ sont tous deux dans RE, alors ils sont tous les deux dans R.
- Démonstration
  - Il existent deux machine de Turing M_L et $M_{\bar{L}}$ qui acceptent respectivement L et $\bar{L}$.
  - Construisons une machine M qui simule parallèlement M_L et $M_{\bar{L}}$. La machine M s'arrête lorsque soit M_L soit $M_{\bar{L}}$ s'arrête.
  - M décide alors bien le langage accepté par M_L, donc $L \in R$ et selon le lemme 2 : $\bar{L} \in R$

Conséquence sur un langage et son complément

- Selon les deux lemmes précédents la position d'un langage et son complément correspond à l'un des trois cas suivant :
  1. L et $\bar{L} \in R$.
  2. $L \notin RE$ et $\bar{L} \notin RE$.
  3. $L \notin RE$ et $\bar{L} \in RE \cap \bar{R}$

Le complément de L0 est dans la classe RE

- Lemme 4 : Le langage défini par :

$$\bar{L_0} = \{w : w = w_i \wedge M_i \text{ accepte } w_i\}$$

  est dans la classe RE.
- Démonstration
  - En TD.
- Théorème 2 : Le langage $\bar{L_0}$ est indécidable.

La technique de réduction

- Technique de démonstration de l'indécidabilité :
- Permet de démontrer l'indécidabilité d'un langage L_2 sachant l'indécidabilté de L_1.
  1. On démontre que s'il existe un algorithme qui décide L_2 alors, il existe aussi un algorithme qui décide L_1.
  2. On donne un algorithme qui décide L_1 en se servant d'un sous programme qui décide L_2. : *On réduit* L_1 *à* L_2.
  3. On conclut que $L_2 \notin R$

La technique de réduction - Exemple

- Le langage universel LU défini par

$$LU = \{< M, w >: M \text{ accepte } w\}$$

  est indécidable
- Démonstration
  - On part du langage $\bar{L_0} = \{w : w = w_i \wedge M_i \text{ accepte } w_i\}$
  - A partir d'un algorithme qui décide LU, Soit w un mot dans $\bar{L_0}$
    1. Déterminer l'indice i tel que $w_i = w$, déterminer M_i
    2. Appliquer l'algorithme de décision pour LU au mot $< M_i, w_i >$ pour décider l'accepter ou de rejeter w_i.

Des problèmes indécidables

- Le problème de l'arrêt et ses variantes
  1. Le problème d'arrêt.
  2. Le problème d'arrêt sur un mot vide.
  3. Le problème d'arrêt existentiel.
  4. Le problème d'arrêt universel
- Problèmes relatifs aux ensembles récursivement énumérables
  1. Déterminer si le langage accepté par une machine de Turing est vide.
  2. Déterminer si le langage accepté par une machine de Turing est récursif.
  3. Déterminer si le langage accepté par une machine de Turing est indécidable.

Propriétés des langages récursivement énumerables

- Théorème 3 : Un langage est calculé par une machine de Turing ssi il est récursivement enumérable.
- Théorème 4 : Un langage est généré par une grammaire ssi il est récursivement enumérable.