Calculabilité
Cours 3 : Problèmes non-calculables

http://www.irisa.fr/lande/pichardie/L3/LOG/

Problèmes et classes de décidabilité

Nous nous intéressons aux problèmes binaires

- question générique dont la réponse est oui/non

Encodage

- chaque instance est codée par un mot sur un alphabet Σ
- *instances positives* : les mots qui encodent une instance pour laquelle la réponse est oui

Un problème est représenté par le langage de ses instances positives

- nous ferons l'amalgame des deux notions dans ce cours

Définitions

Définition (Langages récursifs)

La classe de décidabilité R est l'ensemble des langages décidables par une machine de Turing.

Vocabulaire : la classe R est aussi appelée la classe des langages (problèmes)

- décidés par machine de Turing,
- récursifs , décidables, calculables,
- solubles algorithmiquement.

Définitions

Définition (Langages récursivement énumérables)

La classe de décidabilité RE est l'ensemble des langages acceptés par une machine de Turing.

Vocabulaire : la classe RE est la classe des langages (problèmes)

- acceptés par machine de Turing,
- partiellement récursifs , partiellement décidables, partiellement calculables,
- partiellement solubles algorithmiquement.

Lemme

$$R \subseteq RE$$

Un premier langage indécidable

Est-ce qu'il existe un langage $L \notin R$?

- oui (argument de dénombrement)

Mais nous voulons en exhiber un.

Notations :

- L'ensemble des MT est dénombrable : $M_1, M_2, \ldots$
- L'ensemble des mots (alphabet commun) : $w_1, w_2, \ldots$

Lemme

$$L_0 = \{ w \mid w = w_i \wedge M_i \textit{ n'accepte pas } w_i \}$$

n'est pas dans la classe RE.

Un deuxième langage indécidable

Lemme

Le complément d'un langage de la classe R est un langage de la classe R.

Lemme

Si un langage L et son complément $\overline{L}$ sont tous deux dans RE, alors à la fois L et $\overline{L}$ sont dans R.

Il y a donc trois cas possibles :

1. L et $\overline{L} \in \mathrm{R}$,
2. $L \notin \mathrm{RE}$ et $\overline{L} \notin \mathrm{RE}$,
3. $L \notin \mathrm{RE}$ et $\overline{L} \in \mathrm{RE} \cap \overline{\mathrm{R}}$.

Un deuxième langage indécidable

Lemme

Le langage

$$\overline{L_0} = \{ w \mid w = w_i \wedge M_i \text{ accepte } w_i \}$$

est dans la classe RE.

Théorème

Le langage $\overline{L_0}$ est indécidable (n'appartient pas à R) mais appartient à RE.

La technique de la réduction

Pour montrer que $L_2 \notin$ R, sachant que $L_1 \notin$R,

1. On démontre que s'il existe un algorithme qui décide le langage L_2, alors il existe aussi un algorithme qui décide le langage L_1. Cela se fait en donnant un algorithme (formellement une machine de Turing s'arrêtant toujours) qui décide le langage L_1 en se servant comme d'un sous-programme d'un algorithme décidant L_2.
2. On conclut que L_2 n'est pas décidable ($L_2 \notin$ R) car la réduction de L_1 vers L_2 montre que si L_2 était décidable L_1 le serait aussi, ce qui contredit l'hypothèse que L_1 est un langage indécidable.

Ce type d'algorithme est appelé une réduction de L_1 à L_2. En effet, il réduit la décidabilité de L_1 à celle de L_2.

Exemple

Le *langage universel* LU

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

est indécidable.
(Réduction à partir du langage $\overline{L_0}$)

Des problèmes indécidables

Le problème de l'arrêt

$$H = \{ < M, w > \mid M \text{ s'arrête sur } w \}$$

est indécidable.
(Réduction à partir de LU)

Des problèmes indécidables

Le problème de déterminer si un programme écrit dans un langage de programmation usuel (par exemple en C, ou en Java) s'arrête pour des valeurs fixées de ses données est indécidable. (Réduction à partir du problème de l'arrêt pour les machines de Turing)

Des problèmes indécidables

Le problème de déterminer si une machine de Turing s'arrête lorsque son mot d'entrée est le mot vide (problème de l'arrêt sur mot vide)

$$H_\emptyset = \{ < M > \mid M \text{ s'arrête sur le mot vide } \}$$

est indécidable.
(La réduction se fait à partir du problème de l'arrêt)

Des problèmes indécidables

Le problème de déterminer si le langage accepté par une machine de Turing est vide (langage accepté vide)

$$L_\emptyset = \{ < M > \mid \mathcal{L}(M) = \emptyset \}$$

est indécidable.
(Réduction à partir de $\overline{\text{LU}}$)

Théorème de Rice

Déterminer si le langage accepté par une machine de Turing vérifie une propriété non triviale (i.e. une propriété qui n'est ni vraie pour tout langage, ni fausse pour tout langage) est indécidable
(cf TD)

Les propriétés des langages récursivement énumérables

Les langages récursivement énumérables sont :

- les langages calculés par une machine de Turing,
- les langages énumérés par une procédure effective (ce qui explique l'appellation "récursivement énumérable").

Les langages calculés par une machine de Turing

Définition

Soit une machine de Turing M. Si M s'arrête sur un mot d'entrée u, dénotons par $f_M(u)$ le mot calculé par M pour u. Le langage calculé par M est alors l'ensemble de mots

$$\{ w \mid \exists u \text{ tel que } M \text{ s'arrête pour } u \text{ et } w = f_M(u) \}$$

Théorème

Un langage est calculé par une machine de Turing si et seulement si il est récursivement énumérable (accepté par une machine de Turing).

Les langages énumérables par une procédure effective

Théorème

Un langage est énumérable par une procédure effective si et seulement si il est récursivement énumérable (accepté par une machine de Turing).

Le dixième problème de Hilbert

Trouver un algorithme pour décider si un système d'équations diophantiennes (polynômes à coefficients entiers) a une solution en nombres entiers

Réponse : il n'y a pas de solution, le langage

$$\{\, p \in \mathbb{Z}[X_1, \ldots, X_n] \mid \exists x_1, \ldots, x_n \in \mathbb{Z},\ p(x_1, \ldots, x_n) = 0 \,\}$$

n'est pas décidable.
(Théorème de Youri Matiiassevitch, démontré en 1970)
Question : RE ?

Arithmétique

La question de savoir si oui ou non une proposition énoncée dans le langage de l'arithmétique (avec au moins les deux opérations, $+$ et $\times$) est *vraie* est indécidable.
(c'est une conséquence du premier théorème d'incomplétude de Gödel, ou du théorème de Tarski).

Mais l'arithmétique *de Presburger* (arithmétique entière avec multiplication restreinte au cas où l'un des opérandes est une constante) est décidable.

Que faire après tant de mauvaises nouvelles ?

La vérification automatique de programme pour des propriétés non-triviales est impossible.

Mais. . .

- cela ne vaut pas dire qu'il n'existe pas d'outil de vérification
  - parfois spécialisés pour un certain type de programme
    - vérification du système de vol des airbus A380 par *l'analyseur statique* ASTRÉE
  - parfois demandant l'intervention humaine
    - annotation de *type* dans les programmes
    - preuve interactive avec un *assistant de preuve*

Plan