5. Incalculabilité (suite)

Bruno Grenet

Université Grenoble Alpes – IM²AG
L3 Informatique
UE Modèles de calcul – Machines de Turing

https://membres-ljk.imag.fr/Bruno.Grenet/MCAL-MT.html

Introduction

Incalculabilité par *diagonalisation*

- L'ensemble des algorithmes est dénombrable
- L'ensemble des fonctions est indénombrable

$\Rightarrow$ il existe des fonctions non calculables

Incalculabilité explicite

- Le problème de l'arrêt est indécidable
  - étant donné $\langle A \rangle$ et w, est-ce que $A(w)$ termine ?
- Des variantes sont indécidables
  - est-ce que A s'arrête sur toute entrée ? sur au moins une entrée ? …

Objectifs du jour

1. Théorème de Rice : généralisation du théorème de l'arrêt à *toute propriété* de $L(A)$
2. Problèmes non *reconnaissables*

Rappels

- Langage algos décidable reconnaissables

Table des matières

1. Théorème de Rice

2. Langages non reconnaissables

3. Un peu de recul

Propriété des fonctions

Propriétés et sous-ensembles

- Propriété : « la fonction f ne renvoie que des mots de longueur paire »
- Sous-ensemble $\mathcal{E} = \{f : \forall x \in \{0,1\}^*, |f(x)| \bmod 2 = 0\}$

propriété = ensemble des fonctions qui la satisfont

Définition : propriété *non triviale* d'une fonction calculable

- Sous-ensemble $\mathcal{P}$ des fonctions (partielles) calculables
- $\mathcal{P}$ n'est ni vide, ni l'ensemble de toutes les fonctions calculables

Exemples

- $\mathcal{E}$ est une propriété non triviale : $f_1 : x \mapsto xx$ est dans $\mathcal{E}$, mais $f_2 : x \mapsto x$ ne l'est pas
- « La fonction f est calculable » n'est pas une propriété non triviale :
  - Sous-ensemble des fonctions calculables : ok
  - Trivial : c'est l'ensemble des fonctions calculables

Théorème de Rice

Théorème

Pour toute propriété non triviale $\mathcal{P}$ des fonctions calculables, le problème suivant est indécidable :

Entrée : description $\langle A \rangle$ d'un algorithme

Question : est-ce que la fonction f_A satisfait la propriété $\mathcal{P}$? $f_A \in \mathcal{P}$?

Exemples d'utilisation

- Est-ce que A termine sur l'entrée vide ? $\mathcal{P} = \{f : f(\varepsilon) \downarrow\}$
- Est-ce que A boucle sur au moins une entrée ? $\mathcal{P} = \{f : \exists x, f(x) \uparrow\}$
- Est-ce que A renvoie toujours 0 ? $\mathcal{P} = \{f : \forall x, f(x) = 0\}$
- Est-ce que A ne renvoie que des mots de longueur paire ? $\mathcal{P} = \mathcal{E}$

Preuve du théorème de Rice par *réduction*

Utilisation et limites

Application aux langages

- Étant donné $\langle A \rangle$, est-ce que $L(A) = \{0^n : n \geq 0\}$?
  - Pas directement une question sur f_A
  - Mais presque : est-ce que $f_A(x) = 1 \Leftrightarrow x = 0^n$?

Théorème de Rice pour les langages

Pour toute propriété $\mathcal{P}$ non triviale des langages reconnaissables, étant donné $\langle A \rangle$, déterminer si $L(A) \in \mathcal{P}$ est indécidable.

Attention : exemples de non-utilisation

La question porte sur la fonction f_A ou le langage $L(A)$, pas sur A lui-même

- Est-ce que A a une boucle while ?
- Est-ce que $\mathcal{M}$ a plusieurs rubans ?

Bilan

Théorème de Rice

- Aucune propriété sur la fonction calculée par une machine de Turing n'est décidable
- Aucune propriété sur le langage reconnu par une machine de Turing n'est décidable

Attention !

- On *sait* décider des propriétés sur la machine de Turing elle-même !
  - « la machine possède 12 états »
  - « sur l'entrée w, la machine effectue 100 étapes de calcul »

Généralisation

- Théorème valable pour toute formalisation des algorithmes
- Version langage de programmation :
  - on ne sait *rien* décider sur ce que fait un programme donné
  - on sait décider des caractéristiques du programme lui-même

Table des matières

1. Théorème de Rice

2. Langages non reconnaissables

3. Un peu de recul

Petit rappel

Cours précédent

- Puisque l'ensemble des algorithmes est dénombrable,
  - il existe des fonctions non calculables
  - il existe des langages non décidables
  - il existe des langages non reconnaissables
- Le problème de l'arrêt n'est pas calculable / le langage K n'est pas décidable

Partie précédente de ce cours

- Toutes les propriétés non triviales de f_A et $L(A)$ sont indécidables

Et les langages non reconnaissables alors ? Où sont-ils ?

Un premier résultat positif

Proposition

Le langage $K = \{\langle A, w\rangle : A \text{ termine sur l'entrée } w\}$ est reconnaissable

Preuve

Complémentaire et co-reconnaissable

Définition

- Complémentaire d'un langage $L \subset \{0,1\}^* : \overline{L} = \{w : w \notin L\}$
- Un langage est co-reconnaissable si son complémentaire est reconnaissable
  - Reconnaissable : il existe un algorithme tel que $A(w) = 1 \iff w \in L$
  - Co-reconnaissable : il existe un algorithme tel que $A(w) = 1 \iff w \notin L$

Exemple

- Problème de l'arrêt :
  - $K = \{\langle A, w\rangle : A \text{ termine sur l'entrée } w\}$
  - $\overline{K} = \{\langle A, w\rangle : A \text{ boucle sur l'entrée } w\}$
  - K est reconnaissable, donc $\overline{K}$ est co-reconnaissable

Un deuxième résultat positif

Théorème

Un langage est décidable si et seulement s'il est reconnaissable et co-reconnaissable

Preuve

Une conséquence négative

Corollaire

Le langage $\overline{K} = \{\langle A, w\rangle : A \text{ boucle sur l'entrée } w\}$ n'est pas reconnaissable

Preuve

- Supposons $\overline{K}$ est reconnaissable :
  - Alors K est co-reconnaissable *par définition*
  - Comme K est reconnaissable et co-reconnaissable, il est décidable
  - Contradiction car K n'est pas décidable
- Conclusion : $\overline{K}$ ne peut pas être reconnaissable

Généralisation

- Si L est reconnaissable mais pas décidable, $\overline{L}$ n'est pas reconnaissable

Preuve directe d'un langage non-reconnaissable

Théorème

Le langage $D = \{\langle A \rangle : A \text{ n'accepte pas le mot } \langle A \rangle\}$ n'est pas reconnaissable

Preuve

Et au delà ?

Qu'est-ce qu'on a pour l'instant ?

- Problème de l'arrêt (langage K, fonction H) : reconnaissable / non décidable
- Complémentaire $\overline{K}$, $\overline{H}$: co-reconnaissable / non décidable
- Il existe des langages ni reconnaissables, ni co-reconnaissables
  - Ensemble des algorithmes dénombrable
  - Ensemble des langages indénombrable

Peut-on construire *explicitement* un langage ni reconnaissable, ni co-reconnaissable ?

Fonctions totales

Théorème

Le langage $T = \{\langle A \rangle : A \text{ s'arrête sur toute entrée}\}$ n'est ni reconnaissable, ni co-reconnaissable

Preuve

Bilan

Un dessin

Un théorème de Rice ?

- Il existe une théorème similaire pour les langages non reconnaissables
  - Théorème de Rice-McNaughton-Myhill-Shapiro
  - Trop technique pour ce cours

Table des matières

1. Théorème de Rice

2. Langages non reconnaissables

3. Un peu de recul

Reconnaissable = récursivement énumérable

Définition

- *Récursivement énumérable* : il existe un algorithme qui *énumère* tous les mots de L
  - l'énumération dure un temps infini mais il ne s'écoule qu'un temps fini entre deux mots
  - chaque mot peut être affiché plusieurs fois, et aucun ordre d'énumération n'est fixé
- Formalisation avec une machine de Turing avec un *ruban d'affichage*

Théorème

Un langage est récursivement énumérable si et seulement s'il est reconnaissable

Idée de la preuve

Réductions

Quelles techniques de preuve pour l'indécidabilité/irreconnaissabilité ?

- Preuve directe par *diagonalisation* : si A existe, on tombe sur une contradiction
- Preuve indirecte par *réduction* à un problème indécidable / irreconnaissable :
  - Si L est décidable, alors L' également
  - Or on savait déjà que L' n'était pas décidable, donc L ne l'est pas

Définition

- Une *réduction* (calculable, fonctionnelle) d'un langage L_1 à un langage L_2 est une fonction calculable $f : \Sigma^* \to \Sigma^*$ telle que $w \in L_1 \iff f(w) \in L_2$
- On dit que L_1 est *réductible à* L_2, noté $L_1 \leq_m L_2$

Propriétés

Soit L_1, L_2 tel que $L_1 \leq_m L_2$

- Si L_1 est indécidable, L_2 est indécidable
- Si L_1 est (co-)irreconnaissable, L_2 est (co-)irreconnaissable

Conclusion

Incalculabilité inévitable

- Trop de fonctions / langages par rapport au nombre d'algorithmes
- Constructions explicites
  - Langages non décidables / fonctions non calculables
  - Langages non reconnaissables
  - Langages ni reconnaissables ni co-reconnaissables

Théorèmes de l'arrêt et de Rice

- Aucun algorithme ne peut *expliquer* ce que fait un programme
- Attention !
  - Pas d'algorithme = pas de méthode générale qui marche à tous les coups
  - Mais il est possible de répondre pour *certains* programmes *vérification*

L'incalculabilité dans la *vraie vie* ?

- Théorèmes de l'arrêt et Rice : limite sur les capacités des compilateurs !
- Problèmes indécidables qui ne parlent pas du tout d'algorithmes cours 6