

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
⇒ il existe des fonctions non calculables

Incalculabilité explicite

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

Objectifs du jour

1. Généralisation du théorème de l'arrêt à *toute propriété* de $f_{\mathcal{A}}$ et $L(\mathcal{A})$
2. Problèmes non *reconnaisables*

Rappels

Langages et fonctions

- ▶ Fonction *partielle*: $\{0, 1\}^* \rightarrow \{0, 1\}^*$
- ▶ Langage / problème de décision:
 - ▶ ensemble de mots finis $L \subset \{0, 1\}^*$
 - ▶ fonction *totale* $\chi_L : \{0, 1\}^* \rightarrow \{0, 1\}$

$f(w) \uparrow$: indéfinie sur w

$$\chi_L(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{sinon} \end{cases}$$

Algorithmes

- ▶ Formalisation par machine de Turing, RAM, pseudo-code, ...
- ▶ Algorithme \mathcal{A} :
 - ▶ fonction *partielle* $f_{\mathcal{A}}$ calculée par \mathcal{A}
 - ▶ langage $L(\mathcal{A})$ reconnu par \mathcal{A}

Reconnaissable, décidable, calculable

- ▶ Fonction f calculable: il existe un algorithme \mathcal{A} tel que $f = f_{\mathcal{A}}$ $\mathcal{A}(w) = f(w)$
- ▶ Langage / problème de décision:
 - ▶ L reconnaissable: il existe un algorithme \mathcal{A} tel que $L = L(\mathcal{A})$ $\mathcal{A}(w) = 1 \Leftrightarrow w \in L$
 - ▶ L décidable: il existe un algorithme \mathcal{A} tel $f_{\mathcal{A}} = \chi_L$

Contents

1. Théorème de Rice

2. Langages non reconnaissables

3. Un peu de recul

Propriétés non triviales des fonctions calculables

Définition

- ▶ Propriété des fonctions calculables \Leftrightarrow ensemble de fonctions (qui la satisfont)
- ▶ Propriété non triviale: ni aucune fonction, ni toutes les fonctions

Exemple 1: « la fonction f ne renvoie que des mots de longueur paire »

- ▶ Ensemble $\mathcal{E} = \{f : \forall x \in \{0, 1\}^*, |f(x)| \bmod 2 = 0\}$
- ▶ Propriété non-triviale:
 - ▶ $f_1 : x \mapsto xx$ est dans \mathcal{E}
 - ▶ $f_2 : x \mapsto x$ n'est pas dans \mathcal{E}

Exemple 2: « il existe une machine de Turing \mathcal{M} telle que $f = f_{\mathcal{M}}$ »

- ▶ Ensemble $\mathcal{F} = \{f : f = f_{\mathcal{M}} \text{ pour une machine de Turing } \mathcal{M}\}$
- ▶ Propriété triviale: toute fonction calculable peut l'être par une machine de Turing!
 - ▶ pour tout f calculable, $f \in \mathcal{F}$

Théorème de Rice

Théorème

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

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

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

$f_{\mathcal{A}} \in \mathcal{P} ?$

Exemples d'utilisation

- ▶ Est-ce que $f_{\mathcal{A}}(\varepsilon) \downarrow$?
- ▶ Est-ce qu'il existe x tel que $f_{\mathcal{A}}(x) \uparrow$?
- ▶ Est-ce que $f_{\mathcal{A}}(x) = 0$ pour tout x ?
- ▶ Est-ce que $|f_{\mathcal{A}}(x)|$ est pair pour tout x ?

$$\mathcal{P} = \{f : f(\varepsilon) \downarrow\}$$

$$\mathcal{P} = \{f : \exists x, f(x) \uparrow\}$$

$$\mathcal{P} = \{f : \forall x, f(x) = 0\}$$

$$\mathcal{P} = \mathcal{E}$$

Remarque : propriété non triviale \mathcal{P}

- ▶ Il existe $f_1 \in \mathcal{P}$ et $f_2 \notin \mathcal{P}$
- ▶ Il existe \mathcal{A}_1 et \mathcal{A}_2 tels que $f_{\mathcal{A}_1} \in \mathcal{P}$ et $f_{\mathcal{A}_2} \notin \mathcal{P}$

Preuve du théorème de Rice par réduction

Cas 2 - On sait que $f \notin \mathcal{P}$. Comme \mathcal{P} est non trivial, il existe un algo A^+ tq $f_{A^+} \in \mathcal{P}$.

- Étant donné $\langle A \rangle$ et w , on définit l'algo A_w :

Entrée: y

1. Simuler $A(w)$

2. si $A(w) \downarrow$, simuler A^+ sur l'entrée y

$$\rightarrow f_{A_w} = \begin{cases} f & \text{si } A(w) \uparrow \\ f_{A^+} & \text{sinon} \end{cases} \quad \text{donc } f_{A_w} \in \mathcal{P} \Leftrightarrow A(w) \downarrow.$$

Preuve du théorème de Rice par réduction

On construit un algo H pour le pb de l'arrêt :

Entrée : $\langle A, w \rangle$

1. Calculer $\langle A, w \rangle$

2. Renvoyer $R(\langle A, w \rangle)$

Analyse : $H(\langle A, w \rangle) = R(\langle A, w \rangle) = \begin{cases} 1 & \text{si } \langle A, w \rangle \in \mathcal{P} \\ 0 & \text{sinon} \end{cases} = \begin{cases} 1 & \text{si } A(w) \downarrow \\ 0 & \text{sinon} \end{cases}$

Donc H décide le pb de l'arrêt \rightarrow absurde
donc R n'existe pas.

Théorème de Rice pour les langages

Exemple : étant donné $\langle \mathcal{A} \rangle$, est-ce que $L(\mathcal{A}) = \{0^n : n \geq 0\}$?

- ▶ Pas directement une question sur $f_{\mathcal{A}}$
- ▶ Mais équivalent à : est-ce que $f_{\mathcal{A}}(x) = 1 \Leftrightarrow x = 0^n$?

Théorème de Rice pour les langages

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

Preuve

- ▶ Exactement identique !

Utilisation et limites

Questions sur ce que *fait* un algorithme

- ▶ Est-ce que \mathcal{A} termine sur l'entrée vide ?
- ▶ Est-ce que \mathcal{A} boucle sur au moins une entrée ?
- ▶ Est-ce que \mathcal{A} renvoie toujours 0 ?
- ▶ Est-ce que \mathcal{A} n'accepte que des mots constitués de 0 ?
- ▶ Est-ce que \mathcal{A} accepte tous les mots ?

quoi ?

$$f_{\mathcal{A}}(\varepsilon) \downarrow ?$$

$$\exists x, f_{\mathcal{A}}(x) \uparrow ?$$

$$f_{\mathcal{A}}(x) = 0 \text{ pour tout } x ?$$

$$L(\mathcal{A}) \subset \{0^n : n \geq 0\} ?$$

$$L(\mathcal{A}) = \{0, 1\}^* ?$$

Pas de questions sur le fonctionnement de l'algorithme

comment ?

La question doit porter sur la fonction $f_{\mathcal{A}}$ ou le langage $L(\mathcal{A})$, pas sur \mathcal{A} lui-même

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

Attention aux propriétés triviales !

Il faut une fonction (un langage) dans l'ensemble et un(e) autre hors de l'ensemble

- ▶ Est-ce que $L(\mathcal{A})$ est reconnaissable ?
- ▶ Est-ce que $f_{\mathcal{A}}$ est incalculable ?

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

Contents

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 $f_{\mathcal{A}}$ et $L(\mathcal{A})$ sont indécidables

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

Un premier résultat positif

Proposition

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

Preuve

Algo R: Entrée: $\langle A, w \rangle$

1. Simuler $A(w)$
2. Si $A(w) \downarrow$, renvoyer 1.

$$R(\langle A, w \rangle) = 1 \Leftrightarrow A(w) \downarrow \Leftrightarrow \langle A, w \rangle \in K$$

Complémentaire et co-reconnaissable

Définition

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

Exemple

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

Un deuxième résultat positif

Théorème

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

Preuve

\Rightarrow Si A_0 décide L alors $A_0(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{sinon} \end{cases}$. Donc L est reconnaissable

On construit \bar{A}_0 par $\bar{A}_0(w) = 1 - A_0(w)$. Alors \bar{A}_0 décide \bar{L} , donc \bar{A}_0 reconnaît \bar{L} .

\Leftarrow Soit A_0 qui reconnaît L et \bar{A}_0 qui reconnaît \bar{L} .

On construit D : sur l'entrée w , D simule **en parallèle** A_0 et \bar{A}_0 .
- Dès que l'un renvoie 1, D renvoie 0 ou 1.

D termine car $\begin{cases} \text{si } w \in L, A_0(w) \text{ renvoie } 1 \\ \text{si } w \notin L, \bar{A}_0(w) \text{ renvoie } 1 \end{cases}$.
Parallèle: alternance d'une étape de chaque algo.

Une conséquence négative

Corollaire

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

Preuve

- ▶ Supposons \bar{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 : \bar{K} ne peut pas être reconnaissable

Généralisation

- ▶ Si L est reconnaissable mais pas décidable, \bar{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 $D = \{\langle A \rangle : \langle A \rangle \notin L(A)\}$

- Supposons qu'il existe A_D tq $D = L(A_D)$, tq $A_D(\langle A \rangle) = 1 \Leftrightarrow \langle A \rangle \in D$
- Est-ce que $\langle A_D \rangle \in D$?

$$\langle A_D \rangle \in D \Leftrightarrow A_D(\langle A_D \rangle) = 1 \Leftrightarrow \langle A_D \rangle \notin L(A_D) = D$$

contradiction.

Et au delà ?

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

- ▶ Problème de l'arrêt (langage K , fonction H) : reconnaissable / non décidable
- ▶ Complémentaire \bar{K} , \bar{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-reconnaisable

Preuve

1. T n'est pas co-reconnaisable : $\bar{T} = \{\langle A \rangle : f_A \text{ non totale}\}$ n'est pas reconnaissable

- Étant donné A et w , on définit A_w qui ignore son entrée et simule $A(w)$
- Supp. \bar{T} reconnaissable avec un algo $A_{\bar{T}}$: $A_{\bar{T}}(\langle A \rangle) = 1 \Leftrightarrow f_A \text{ non totale}$.

- On construit \bar{H} :

Entrée: $\langle A, w \rangle$

1. Calcule $\langle A_w \rangle$

2. Renvoie $A_{\bar{T}}(\langle A_w \rangle)$

$\bar{H}(\langle A, w \rangle) = A_{\bar{T}}(\langle A_w \rangle)$. Donc

$\bar{H}(\langle A, w \rangle) = 1 \Leftrightarrow f_{A_w} \text{ non totale} \Leftrightarrow A(w) \uparrow$

Donc \bar{H} reconnaît \bar{K} : absurde donc $A_{\bar{T}}$ n'existe pas.

Fonctions totales

Théorème

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

Preuve

2. T n'est pas reconnaissable : on veut la même chose avec A'_w
 t_1 $f_{A'_w}$ totale $\Leftrightarrow A(w) \uparrow$

A'_w : Entrée : t

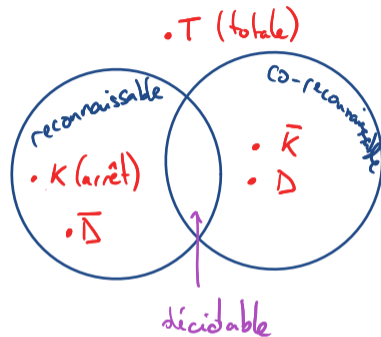
1. Simuler $A(w)$ pendant $\leq t$ étapes
2. si $A(w)$ a terminé : boucle infinie
3. sinon : renvoyer 1

$f_{A'_w}$ totale
 \Updownarrow
 $\forall t, A(w)$ ne termine pas en $\leq t$ étapes
 \Updownarrow
 $A(w) \uparrow$

\rightarrow fin identique à la partie 1.

Bilan

Un dessin



Un théorème de Rice ?

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

Contents

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

① ⇒ Sur une entrée w , on énumère tous les mots de L jusqu'à voir w .

- Si $w \in L$, on va le trouver et on répond oui
- Si $w \notin L$, on boucle.

② ⇐ Pour chaque mot $w \in \{0,1\}^*$, on lance la reconnaissance (en parallèle)
↳ si le mot est reconnu, on l'affiche.

Réductions

Quelles techniques de preuve pour l'indécidabilité/irreconnaissabilité ?

- ▶ Preuve directe par *diagonalisation*: si \mathcal{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 $f : \Sigma^* \rightarrow \Sigma^*$ 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 *certain*s 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