4. Incalculabilité

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

Thèse de Church-Turing

Tout ce qui est *calculable* est calculable par machine de Turing

Rappels

- Une fonction f est calculable si $f = f_{\mathcal{M}}$ pour une machine de Turing $\mathcal{M}$
- Un langage L est $\begin{cases} \text{reconnaissable si } L = L(\mathcal{M}) \text{ pour une machine de Turing } \mathcal{M} \\ \text{décidable si } \chi_L \text{ est calculable} \end{cases}$

- Il existe des fonctions *incalculables* ? des langages *irreconnaissables* ? *indécidables* ?
- Qu'est-ce que ça veut dire ?

Contents

Contents

Pas assez de machines de Turing !

Il y a *moins* de machines de Turing que de langages, et que de fonctions

Qu'est-ce que ça veut dire ?

- Il y a une infinité de machines de Turing
- Il y a une infinité de langages
- Il y a une infinité de fonctions

Différentes tailles d'infini

- Ensemble dénombrable X : définitions équivalentes
  - il existe une fonction *injective* $n : X \to \mathbb{N}$ $\quad x \neq y \Leftrightarrow n(x) \neq n(y)$
  - on peut *numéroter* les objets de X $\quad x$ reçoit le numéro $n(x)$
  - chaque $x \in X$ possède une description finie

  Exemples : $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{Z}$, $\{0,1\}^*$, Σ^*, $Q \times \Sigma^*$, ...
- Ensemble indénombrable : ensemble non dénombrable

  Exemples : $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}^{\mathbb{N}}$: suites $(u_n)_{n \in \mathbb{N}}$, $\{f : \mathbb{R} \to \mathbb{R}\}$, ...

L'infini des machines de Turing

Théorème

- L'ensemble des machines de Turing est dénombrable
  - L'ensemble des algorithmes est dénombrable

Rappel

- $\mathcal{M} = (Q, \Sigma, q_0, \delta)$
- Il suffit de représenter la table δ avec cinq colonnes

Preuve

- $\mathcal{M}$ est entièrement décrite par la table δ
- Une table est une suite finie de symboles $\rightsquigarrow$ description finie.

L'infini des langages

Théorème

L'ensemble $\mathcal{L} = \{L : L \subset \{0,1\}^*\}$ de tous les langages est **indénombrable**

Intuition

- Ensemble $\mathcal{L}$ = ensemble d'ensembles infinis — L = ensemble infini de mots
- Pas de représentation finie de chaque langage L
  - Il faut lister tous les mots
  - Exemple : $L = \{\varepsilon, 0, 1, 00, 11, 000, 010, 101, 111, 0000, 0110, 1001, \dots\}$ — *palindromes*

Remarque

Ce n'est pas une preuve !

- On ne *voit pas* comment représenter chaque langage (infini) de manière finie
- On va *prouver* que ce n'est effectivement pas possible

Langages et suites binaires *infinies*

Numérotation de $\{0,1\}^*$

- $\{0,1\}^*$ est dénombrable $\rightarrow$ on peut *numéroter* les mots
- Exemple : $n(x) = \overline{1x}^2 - 1$ sinon — écriture en base 2
  - $n(\varepsilon) = 0$, $n(0) = 1$, $n(1) = 2$, $n(00) = 3$, …

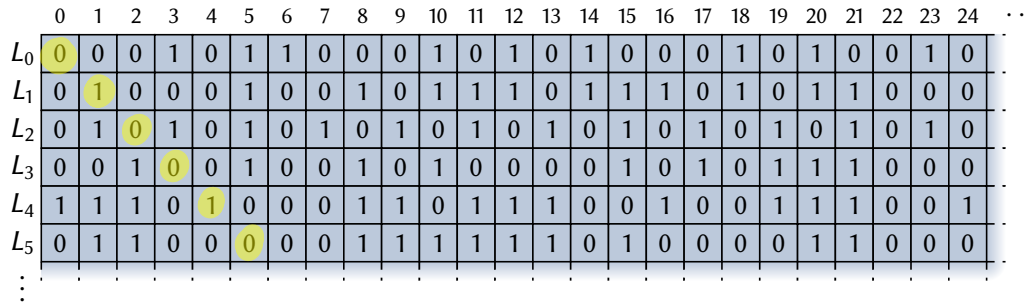
Langage $\simeq$ suite binaire infinie

- Langage $L \rightsquigarrow$ suite $(w_i)_{i \geq 0}$ infinie sur $\{0,1\}$:
  - si $x \in L$, $w_{[n(x)]} = 1$
  - si $x \notin L$, $w_{[n(x)]} = 0$
- Exemple : $L = \{\varepsilon, 0, 1, 00, 11, 000, 010, 101, 111, 0000, 0110, 1001, \dots\}$

x	ϵ	0	1	00	01	10	11	000	001	010	011	100	101	110	111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
$n(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$w =$	1	1	1	1	0	0	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	0	0	1

Preuve par l'absurde : *diagonale de Cantor*

- On suppose $\mathcal{L}$ dénombrable : on peut numéroter les langages $L_0, L_1, \ldots$
- On construit $L \notin \mathcal{L}$: contradiction $\qquad \forall i, L \neq L_i$



$\forall i,$ le mot n°i est dans L_i $\Leftrightarrow$ le mot n°i n'est pas dans L

Donc $L \neq L_i$

Conséquence

Deux faits

- L'ensemble des algorithmes est dénombrable algorithme=machine de Turing
- L'ensemble des langages (resp. des fonctions) est indénombrable

Théorème

- Il existe des langages $L \subset \{0,1\}^*$ qui ne sont pas reconnaissables
- Il existe des langages $L \subset \{0,1\}^*$ qui ne sont pas décidables
- Il existe des fonctions $f : \{0,1\}^* \to \{0,1\}^*$ qui ne sont pas calculables

Preuve

- On peut numéroter les algorithmes : $A_0, A_1, \ldots$
- Si pour chaque langage L, on associe un algorithme A_i → la diagonale de Cantor montre qu'il existe forcément un langage sans algorithme.

Bilan

Plusieurs tailles d'infini

- Ensembles dénombrables ou indénombrables
- Il existe même une infinité de tailles d'infini différentes

Tout n'est pas calculable

- Il existe beaucoup plus de langages ou de fonctions que d'algorithmes
- Donc il existe des langages et fonctions sans algorithme

Mais ils sont où ?

- Le théorème ne donne *aucun exemple* !

Contents

Algorithmes, fonctions, langages, problèmes

$\Sigma = \{0,1\}$

Vocabulaire (rappel)

- Fonction (partielle) ou problème $f : \Sigma^* \to \Sigma^*$
- Langage : $L \subset \Sigma^*$
- Algorithme :
  - soit définition informelle et intuitive
  - soit objet dans une formalisation possible — machine de Turing, λ-calcul, …

Problèmes de décision et langages

- Problème de décision : fonction *totale* $f : \Sigma^* \to \{0,1\}$
- Équivalent à un langage : $f \leftrightarrow L_f = \{w : f(w) = 1\}$

Dans la suite

- On ne s'intéresse qu'à des *problèmes de décision/langages*
  - On utilise les deux formalismes en fonction de ce qui est le plus pratique
  - Formalisation possible : machines de Turing avec un état « oui »

Programmes et données

Un programme est une donnée comme un autre !

- Programme dans un langage donné : fichier texte
- Avec un interpréteur : exécution du programme contenu dans le fichier
- Mais aussi : programmes qui modifient d'autres programmes

Machines de Turing

- Rappel :
  - Machine $\mathcal{M}$ représentée par une table à cinq colonnes
  - Codage binaire $\langle \mathcal{M} \rangle \in \{0,1\}^*$ d'une machine
- Une machine $\mathcal{N}$ peut prendre en entrée le code d'une machine $\langle \mathcal{M} \rangle$

Autres formalisations

- Exactement pareil avec des machines RAM, des automates cellulaires, …

Un algorithme peut prendre en entrée (la description d')un algorithme

Machine universelle

Définition

Une machine de Turing $\mathcal{U}$ est dite universelle si sur l'entrée $\langle \mathcal{M}, x \rangle$, $\mathcal{U}$ *simule le comportement* de $\mathcal{M}$ sur l'entrée x :

- $\mathcal{U}$ termine sur l'entrée $\langle \mathcal{M}, x \rangle$ si et seulement si $\mathcal{M}$ termine sur l'entrée x
- Si $\mathcal{U}$ termine, elle renvoie la même valeur que $\mathcal{M}$

Théorème fondamental

Il existe une machine de Turing universelle

Idée de la preuve

Voir exemple

Utilités théorique et pratique

Théorie

- Dans tout modèle Turing-complet, il existe un algorithme universel
  - Un algorithme universel permet de calculer toute fonction calculable
- Un modèle est Turing-complet s'il peut simuler une machine universelle

- Une machine $\mathcal{N}$ peut prendre $\langle \mathcal{M} \rangle$ en entrée *et simuler* $\langle \mathcal{M} \rangle$ *sur l'entrée de son choix*

Pratique

- Tout langage de programmation admet un *programme universel*
  - interpréteur / compilateur écrit dans le langage lui-même
- En quel langage est écrit GCC ?

Dans la suite de ce cours

Algorithme = machine de Turing (ou autre…)

- Formalisation implicite
- Description des algorithmes en *pseudo-code*
  - traduction possible en machine de Turing ou autre formalisation
  - pseudo-code extrêmement simple → pas de triche !
- Pour un algorithme $\mathcal{A}$, $\langle \mathcal{A} \rangle$ est sa description binaire

Conséquences de la machine universelle

- Étant donné $\langle \mathcal{A} \rangle$ et w, un algorithme peut simuler $\mathcal{A}(w)$
  - Ex.: si $\mathcal{A}(w) = 1$, renvoyer 0 ; si $\mathcal{A}(w) = 0$, rentrer dans une boucle infinie
- Étant donné $\langle \mathcal{A} \rangle$, un algorithme peut calculer une variante $\langle \mathcal{A}' \rangle$ de $\langle \mathcal{A} \rangle$
  - Ex.: étant donné $\langle \mathcal{A} \rangle$ et w, calculer $\langle \mathcal{A}_w \rangle$ qui ignore son entrée et exécute $\mathcal{A}(w)$

→ Difficultés (éventuellement) techniques mais pas conceptuelles

Contents

Définition

Problème de l'arrêt (informel)

Entrées : un algorithme $\mathcal{A}$
une entrée w
Sortie : 1 si $\mathcal{A}$ termine sur l'entrée w
0 si $\mathcal{A}$ ne termine pas sur l'entrée w

Formalisations

- Fonction H : $\{0,1\}^* \to \{0,1\}$

$$\langle \mathcal{A}, w \rangle \mapsto \begin{cases} 1 & \text{si } \mathcal{A} \text{ termine sur l'entrée } w \\ 0 & \text{sinon} \end{cases}$$

- Langage $K = \{\langle \mathcal{A}, w \rangle : \mathcal{A} \text{ termine sur l'entrée } w\} \subset \{0,1\}^*$

Incalculabilité

Théorème de l'arrêt

Le problème de l'arrêt n'est pas calculable

Preuve

. On suppose qu'il existe A_H tq $A_H(\langle A, w\rangle) = \begin{cases} 1 & \text{si } A(w)\downarrow \\ 0 & \text{si } A(w)\uparrow \end{cases}$

. On construit Δ :

Entrée : $\langle A\rangle$

1. Si $A_H(\langle A, A\rangle) = 1$, boucle infinie
2. Sinon : renvoyer 1.

. Que fait Δ sur l'entrée $\langle\Delta\rangle$?

- Si $A_H(\langle\Delta,\Delta\rangle) = 1$, $\Delta(\langle\Delta\rangle)$ boucle
- Sinon : $\Delta(\langle\Delta\rangle)$ termine.

} A_H s'est "trompé" sur l'entrée $\langle\Delta,\Delta\rangle$.

$\Rightarrow$ A_H ne peut pas exister

D'autres problèmes incalculables

Corollaire

Les langages suivants sont indécidables :

- $\{\langle \mathcal{A} \rangle : \mathcal{A}$ termine sur l'entrée vide$\}$ $=: L_1$
- $\{\langle \mathcal{A} \rangle : \mathcal{A}$ termine sur au moins une entrée$\}$
- $\{\langle \mathcal{A} \rangle : \mathcal{A}$ termine sur toute entrée$\}$
- $\{\langle \mathcal{A} \rangle : \mathcal{A}$ renvoie toujours 1$\}$
- …

versions fonction

est-ce que $\mathcal{A}(\varepsilon) \downarrow$?

est-ce qu'il existe x t.q. $\mathcal{A}(x) \downarrow$?

est-ce que $\mathcal{A}(x) \downarrow$ pour tout x ?

est-ce que $\mathcal{A}(x) = 1$ pour tout x ?

Preuve du premier

- On suppose L_1 décidable avec un algo A_{D_1} : $A_{D_1}(\langle A \rangle) = \begin{cases} 1 \text{ si } A(\varepsilon)\downarrow \\ 0 \text{ si } A(\varepsilon)\uparrow \end{cases}$
- On construit A_H :

  Entrée : $\langle A, w \rangle$

  1. Calcule $\langle A_w \rangle$ qui ignore son entrée et simule $A(w)$
  2. Renvoyer $A_{D_1}(\langle A_w \rangle)$.

- Alors $A_H(\langle A, w \rangle) = A_{D_1}(\langle A_w \rangle) = \begin{cases} 1 \text{ si } A_w(\varepsilon)\downarrow \\ 0 \text{ sinon} \end{cases} = \begin{cases} 1 \text{ si } A(w)\downarrow \\ 0 \text{ sinon.} \end{cases}$

Donc A_H résout le problème de l'arrêt → impossible

Conclusion

Il existe des langages / fonctions incalculables

- Pas assez d'algorithmes pour tous les langages ou toutes les fonctions
- Aucun algorithme ne peut décider, étant donné $\langle \mathcal{A} \rangle$ et w, si $\mathcal{A}(w)$ termine

Quelques conséquences

- Variantes également indécidables : terminaison sur l'entrée vide, sur toute entrée, …
- Attention !
  - Pas d'algorithme = pas de méthode générale qui marche à tous les coups
  - Mais il est possible de répondre pour *certains* programmes *vérification*
- Mêmes résultats dans tout modèle (automates cellulaires, λ-calcul, …)
  - Même en croisant : une machine de Turing ne peut décider l'arrêt d'une machine RAM

Allons plus loin !

- Aucune *propriété* de la fonction calculée par $\mathcal{A}$ n'est décidable *cours 5*
- Il existe des langages non reconnaissables *cours 5*
- Il existe des problèmes indécidables qui ne parlent pas d'algorithmes *cours 6*