

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 $\begin{cases} \text{reconnaisable si } L = L(\mathcal{M}) \text{ pour une machine de Turing } \mathcal{M} \\ \text{décidable si } \chi_L \text{ est calculable} \end{cases}$

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

Table des matières

1. Dénombrable et indénombrable

2. Machine universelle

3. Le problème de l'arrêt

Table des matières

1. Dénombrable et indénombrable

2. Machine universelle

3. Le problème de l'arrêt

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* $n : X \rightarrow \mathbb{N}$
 - ▶ on peut *numéroter* les objets de X
 - ▶ chaque $x \in X$ possède une description finie

Exemples: $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \{0,1\}^*, \Sigma^*, \mathbb{Q} \times \Sigma^*, \dots$

- ▶ Ensemble indénombrable: ensemble non dénombrable

Exemples: $\mathbb{R}, \{f: \mathbb{R} \rightarrow \mathbb{R}\}, \mathbb{Z}^{\mathbb{N}}, \dots$

$x \neq y \Leftrightarrow n(x) \neq n(y)$
 x reçoit le numéro $n(x)$

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

- Pour décrire Ω , il suffit de décrire \mathcal{S}
- Représentation de \mathcal{S} finie en tableau à 5 colonnes $\rightsquigarrow \{0,1\}^*$.

L'infini des langages

Théorème

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

Idée de la preuve

- ▶ 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*

Représentation plus compacte

- ▶ $\{0, 1\}^*$ est dénombrable \rightarrow on peut *numéroter* les mots exemple : $n(x) = \overline{1x}^2 - 1$
- ▶ Langage $L \rightsquigarrow$ mot w **infini** sur $\{0, 1\}$ $w_{[i]} = 1 \Leftrightarrow$ mot $n^\circ i$ dans L

$n(x)$	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	
$w =$	1	1	1	1	0	0	1	1	0	1	0	0	1	0	1	1	0	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, \dots

► On construit $L \notin \mathcal{L}$: contradiction

$$\forall i, L \neq L_i$$

	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	...
L_0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	1	0	
L_1	0	1	0	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	1	0	1	1	0	0	0	
L_2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
L_3	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	1	0	1	1	1	0	0	0	
L_4	1	1	1	0	0	0	0	0	1	1	0	1	1	1	0	0	1	0	0	1	1	1	0	0	1	
L_5	0	1	1	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	1	1	0	0	0	
⋮																										

L	1	0	1	1	0	1	.	.	.																	
-----	---	---	---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

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 $f : \{0,1\}^* \rightarrow \{0,1\}^*$ qui ne sont pas calculables

Preuve

- On peut numéroter les algorithmes A_0, A_1, \dots
- Si à chaque algo, on associe un langage L , alors il existe des langages 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!*

Table des matières

1. Dénombrable et indénombrable

2. Machine universelle

3. Le problème de l'arrêt

Vocabulaire (rappel)

- ▶ Fonction (partielle) ou **problème** $f : \Sigma^* \rightarrow \Sigma^*$
- ▶ 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** $f : \Sigma^* \rightarrow \{0, 1\}$
- ▶ 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

- 4 rubans : $\langle \Gamma_b \rangle$; q ; $\langle x \rangle$; Travail (ruban de $\langle \Gamma_b \rangle$)

état courant

- Transition $\delta(q, x) = (q', y, \leftarrow)$ de Γ_b :

1. Aller lire $\delta(q, x)$ dans $\langle \Gamma_b \rangle$ (1^{er} ruban)
2. Modifier $q \rightarrow q'$ (2^e ruban)
3. Modifier $x \rightarrow y$ (4ⁱ ruban)
4. Déplacer la tête (—)

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 A , $\langle A \rangle$ est sa description binaire

Conséquences de la machine universelle

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

→ Difficultés (éventuellement) techniques mais pas conceptuelles

Table des matières

1. Dénombrable et indénombrable

2. Machine universelle

3. Le problème de l'arrêt

Définition

Problème de l'arrêt (informel)

Entrées : un algorithme A
une entrée w

Sortie : 1 si A termine sur l'entrée w
0 si A ne termine pas sur l'entrée w

Formalisations

▶ Fonction $H : \{0, 1\}^* \rightarrow \{0, 1\}$

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

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

Incalculabilité

Théorème de l'arrêt

Le problème de l'arrêt n'est pas calculable

Preuve

- On suppose qu'il existe A_{H} qui calcule $H(\langle A, x \rangle) = \begin{cases} 1 & \text{si } A(x) \downarrow \\ 0 & \text{si } A(x) \uparrow \end{cases}$
- On construit un nouvel algo D :
Entrée: $\langle A \rangle$
 1. Si: $A_{\text{H}}(\langle A, A \rangle) = 1$: rentrer dans une boucle infinie
 2. Sinon: renvoyer 1.
- Analyse: que fait $D(\langle D \rangle)$?
 - si $A_{\text{H}}(\langle D, D \rangle) = 1$, alors boucle infinie: $D(\langle D \rangle) \downarrow \Rightarrow D(\langle D \rangle) \uparrow$
 - si $A_{\text{H}}(\langle D, D \rangle) = 0$, alors renvoie 1: $D(\langle D \rangle) \uparrow \Rightarrow D(\langle D \rangle) \downarrow$

D'autres problèmes incalculables

Corollaire

Les langages suivants sont indécidables :

- ▶ $\{\langle A \rangle : A \text{ termine sur l'entrée vide}\} = L_\varepsilon$
- ▶ $\{\langle A \rangle : A \text{ termine sur au moins une entrée}\}$
- ▶ $\{\langle A \rangle : A \text{ termine sur toute entrée}\}$
- ▶ $\{\langle A \rangle : A \text{ renvoie toujours 1}\}$
- ▶ ...

versions fonction

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

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

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

est-ce que $A(x) = 1$ pour tout x ?

Preuve du premier

- Supposons que L_ε soit décidable avec un algo $T : T(\langle A \rangle) = \begin{cases} 1 & \text{si } A(\varepsilon) \downarrow \\ 0 & \text{si } A(\varepsilon) \uparrow \end{cases}$

- On construit A_{0H} : Entrée : $\langle A, x \rangle$

1. Calcule $\langle A, x \rangle$ de l'algo qui ignore son entrée et simule $A(x)$

2. Renvoie $T(\langle A, x \rangle) \Rightarrow A_{0H}(\langle A, x \rangle) = \begin{cases} 1 & \text{si } A(x) \downarrow \\ 0 & \text{si } A(x) \uparrow \end{cases}$

- Conclusion : puisque A_{0H} ne peut pas exister, T non plus.

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 $\langle A \rangle$ et w , si $A(w)$ 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 *certaines* 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 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*