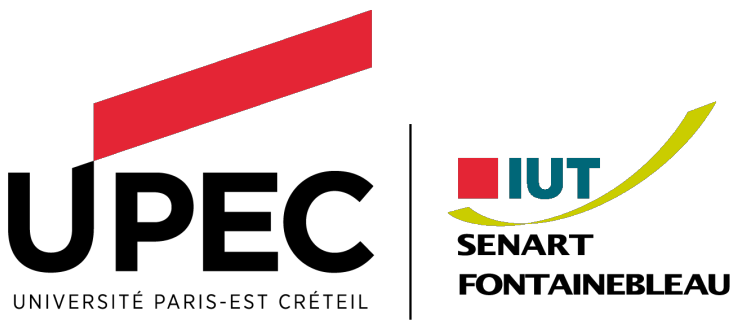
Fondements de l’informatique

Indécidabilité

Florent Madelaine

Fondements de l’informatique



Plan

Introduction

On a vu divers modèles :

- automate
- automate à pile
- Machine de Turing

La thèse de Church Turing

Informellement

Pour tout modèle raisonnable de calcul, on obtient la même notion de ce qui est calculable.

Arguments en faveur de cette thèse

- Théorie générales des fonctions récursives (Gödel, Herbrand 1933)
- λ-calcul (Church, 1936)
- Machines de Turing (Turing, 1936)
- Trois modèles équivalents (Church 1936, Turing 1937).
- Machines de Post (1936)
- Machines de Turing avec plusieurs rubans (Minsky)
- Machines à compteurs
- Machines à registre
- ...

La thèse de Church Turing

Informellement

Pour tout modèle raisonnable de calcul, on obtient la même notion de ce qui est calculable.

Arguments en faveur de cette thèse

- Théorie générales des fonctions récursives (Gödel, Herbrand 1933)
- λ-calcul (Church, 1936)
- Machines de Turing (Turing, 1936)
- Trois modèles équivalents (Church 1936, Turing 1937).
- Machines de Post (1936)
- Machines de Turing avec plusieurs rubans (Minsky)
- Machines à compteurs
- Machines à registre
- ...

Vocabulaire

Définition

Un langage L est décidable si il existe une machine de Turing M qui accepte exactement les mots de ce langage, c'est-à-dire que pour tout mot x sur son ruban la machine M arrête son calcul dans un état acceptant ssi x est dans L.

Définition

Une fonction f est calculable si il existe une machine de Turing M qui étant donné x sur son ruban s'arrête avec comme contenu de ruban $f(x)$.

Un peu de dénombrement

- L’ensemble des programmes est dénombrable
- L’ensemble des langages n’est pas dénombrable

Limite forte de ce qu’on peut calculer

Il existe « beaucoup » de langages pour lesquels nous n’avons pas de programme.

Plus de détails sont donnés dans la suite.

On peut dénombrer les programmes

Lemme

L'ensemble des programmes est dénombrable

On peut rester informel comme ci-dessus et parler de langage de programmation quelconque (du C par exemple) et via le codage ascii faire correspondre au texte du programme un nombre en binaire. La croyance en la thèse de Church-Turing faisant le reste.

Alternativement si on souhaite être plus précis on peut montrer ceci.

Lemme

L'ensemble des machines de Turing est dénombrable.

On peut dénombrer les programmes

Lemme

L'ensemble des machines de Turing est dénombrable.

Démonstration.

On peut coder en binaire les états, les symboles de l'alphabet et les actions de déplacement de la tête de lecture.
À l'aide de symboles séparateurs adaptés (parenthèses, virgules), on peut lister les transitions. Par exemple une liste de mots de la forme `(état avant, symbole lu, état après, symbole écrit, déplacement)`.
On peut ensuite reprendre l'argument précédent avec le code ascii. □

Digression : code d'une machine de Turing

On va appeler code d'une machine de Turing le mot suivant : `nombre d'états en binaire , nombre de symboles en binaire , liste des transitions dans la preuve précédente séparées par des virgules`.
Quitte à ajouter des 0 à gauche, on fait en sorte que tous les mots binaires codant des états, des symboles ou des actions ont la même longueur.

Notez que ce mot est sur l'alphabet comportant les symboles `0 1 , ( )`

Digression : configuration d'une machine de Turing

Quand on travaille sur les MdT il est pratique de pouvoir décrire « la photo »de la machine à un instant du calcul. On parle de configuration : il s'agit du contenu du ruban, de la position de la tête et de l'état.

Dans l'esprit de notre code, nous pouvons donner un code pour une configuration. `codes des lettres à gauche de la tête , code de l'état , codes des lettres à droite de la tête .`

Notez que ce mot est sur l'alphabet comportant les symboles `0 1 ,`

Retour à nos moutons : trop de langages

Lemme

L'ensemble des langages sur l'alphabet $\{0, 1\}$ *n'est pas dénombrable.*

Démonstration.

On énumère les mots binaires dans l'ordre lexicographique $s_0 = 0, s_1 = 1, s_2, 01, s_3 = 10, s_4 = 11, \ldots$.

On suppose par l'absurde que l'ensemble de tous les langages sur $\{0, 1\}$ est dénombrable et donc qu'on peu énumérer sous forme de liste $L_0, L_1, L_2, \ldots$ (l'index est donné par la bijection de $\mathbb{N}$ dans l'ensemble de tous les langages).

Soit L le langage *diagonal* : $\{s_i$ tel que $s_i \notin L_i\}$.

L apparaît dans la liste, donc il existe un index j pour lequel $L = L_j$. Ceci est absurde puisque $s_j \in L \iff s_j \notin L_j$. □

Indécidabilité

Théorème

Il existe des langages sur l'alphabet $\{0, 1\}$ *qui ne sont pas décidables (par une machine de Turing).*

Nous allons voir dans la suite qu'on peut exhiber un exemple concret : le problème de l'arrêt d'un programme.

Machine de Turing Universelle

On a vu qu'on peut coder (le programme) d'une machine de Turing M comme un mot sur l'alphabet comportant les symboles `0 1 , ( )`

On peut coder l'entrée x d'une telle machine par un mot dans le même esprit.

La machine de Turing universelle U prend sur son ruban le code d'une machine M, suivi d'un `;` suivi du code d'une entrée x. Cette machine universelle travaille donc sur l'alphabet comportant les symboles `0 1 , ( ) ;`

Notation simplifiée pour l'entrée : on écrira juste `M;x`

Programme de la machine de Turing Universelle

En gros la machine va travailler sur x en utilisant le programme M. Le principe est similaire à la manière dont un programme assembleur est traité par un processeur.

Pour décrire proprement le fonctionnement de cette machine il est plus facile de considérer une machine de Turing avec plusieurs rubans (on pourra toujours la simuler par une machine à un seul ruban dans un second temps).

Programme de la machine de Turing Universelle

Le premier ruban reste initialement inchangé. Le second ruban contient le code de la configuration de la machine M. La machine U simule le calcul de M pas à pas : scan de l'état actuel de M sur le second ruban, recherche d'une règle adaptée sur le premier ruban, changement adapté de la configuration sur le second ruban.

Si l'entrée de U est incohérente et ne correspond pas au code d'une machine de Turing, la machine U déplace sa tête indéfiniment vers la droite.

Un exemple concret de problème indécidable

Définition (problème de l'arrêt)

$$H = \{M; x \text{ tel que M s'arrête sur l'entrée x}\}$$

($M; x$ est codé comme expliqué précedemment)

Théorème

Le problème de l'arrêt est indécidable.

La preuve se fait par l'absurde avec un argument de diagonalisation.

Preuve

Par l'absurde. Soit M_H une machine qui décide H.

Soit D une machine qui prend en entrée le code d'une machine de Turing M. D accepte M ssi M_H ne s'arrête pas sur l'entrée $M;M$.

La contradiction apparaît lorsqu'on se penche sur le calcul de la machine D sur son propre code en entrée.

Réduction de Turing et indécidabilité d'autres problèmes

On peut utiliser le concept de réduction pour montrer que d'autres problèmes sont indécidables.

Exemple

$$S := \{M \text{ tel que } M \text{ s'arrête sur toute entrée}\}$$

Si S était décidable, on pourrait y réduire le problème H (détails ci-dessous).

- Entrée de H : $M; x$.
- Construction de la machine M' qui va prendre en entrée y et s'arrêter si y est différent de x, sinon en cas d'égalité la machine M' répond comme M sur l'entrée x.
- M' appartient à S ssi $M; x$ appartient à H.

Hiérarchie

Nous avons entrevu le modèles de Turing. Nous avons vu qu'il permet de décider un langage qui n'est pas régulier.

Nous avons argumenté que tout ce qui peut se calculer peut l'être avec une MdT (même si ce n'est pas pratique pour programmer en pratique évidemment).

Nous avons vu que les problèmes sont beaucoup plus nombreux que les programmes (cardinalité du discret vs continu) en particulier le problème de l'arrêt est indécidable.

Pour aller plus loin

Le théorème de Rice indique que toute propriété sur les machines sont indécidables à condition d'être non triviale.

On peut en déduire par exemple qu'il est impossible de savoir si une MdT est équivalente à un automate.

Il est possible de dépasser l'indécidable en acceptant les machines à Oracle.