Chapitre 2 : Calculabilité et fonctions récursives

ENSIIE - Modèles de calculs - Machines de Turing

Dimitri Watel (dimitri.watel@ensiie.fr)

2018

Remarque : dans tout ce cours, on parle de Machines déterministes.

Fonctions primitives récursives

Définition : $\mathcal{F}_p$ et $\mathcal{F}$

On note $\mathcal{F}_p$ l'ensemble des fonctions de $\mathbb{N}^p \to \mathbb{N}$, pour $p \in \mathbb{N}$. On identifie $\mathcal{F}_0 = \mathbb{N}$.

Ces fonctions sont totales, définies sur tout N^p

$$\mathcal{F} = \bigcup_{p \in \mathbb{N}} \mathcal{F}_p$$

Fonction primitive récursive

Une fonction primitive récursive est une fonction de $\mathcal{F}$ construite selon un procédé récursif simple :

- trois fonctions *initiales*
- une fonction de *composition* de fonctions
- une fonction de *récursion primitive* de fonctions

Fonctions primitives récursives : fonctions initiales

Fonction **0**

$$\mathbf{0} : x \to 0$$

0 est une fonction de $\mathcal{F}_0$.

Fonction Successeur σ

$$\sigma : x \to x + 1$$

σ est une fonction de $\mathcal{F}_1$.

Fonction Projection $\mathcal{P}_{i,p}$

Soient $p \in \mathbb{N}$ et $i \leq p$.

$$\mathcal{P}_{i,p} : x_1, x_2, \ldots, x_p \to x_i$$

$\mathcal{P}_{i,p}$ est une fonction de $\mathcal{F}_p$.

Fonctions primitives récursives : composition

Fonction *composition*

Soient $p, n \in \mathbb{N}$.

$$\circ_{n,p} : \mathcal{F}_n \times \underbrace{\mathcal{F}_p \times \cdots \times \mathcal{F}_p}_{n} \to \mathcal{F}_p$$

$\circ$ n'est pas une fonction à valeur entière mais une fonction de fonctions. Elle permet de construire des fonctions à partir d'autres fonctions. On note $\circ(g, f_1, f_2, \ldots, f_n)$ ou $g \circ (f_1, f_2, \ldots, f_n)$ ou $g(f_1, f_2, \ldots, f_n)$.
Posons $x = (x_1, x_2, \ldots, x_p)$.

$$g \circ (f_1, f_2, \ldots, f_n) : x \to g(f_1(x), f_2(x), \ldots, f_n(x))$$

On omettra n et p près du symbole $\circ$ sauf abiguïté.

Fonctions primitives récursives : récursion primitive

Fonction *récursion primitive*

Soit $p \in \mathbb{N}$.

$$\circlearrowleft_p : \mathcal{F}_p \times \mathcal{F}_{p+2} \to \mathcal{F}_{p+1}$$

Comme pour $\circ$, $\circlearrowleft$ est une fonction de fonctions.
Posons $x = (x_1, x_2, \ldots, x_p)$ et $f = \circlearrowleft (g, h)$

$$f(x, n) \to \begin{cases} g(x) & \text{si } n = 0 \\ h(x, n-1, f(x, n-1)) & \text{sinon} \end{cases}$$

On omettra p près du symbole $\circlearrowleft_p$ sauf abiguïté.

sert à paramétrer l'itération $(n-1)$

Fonctions primitives récursives

"le plus petit"

Fonctions primitives récursives : $\mathcal{PR}$

L'ensemble $\mathcal{PR}$ est l'ensemble des fonctions de $\mathcal{F}$ telles que :

- $\mathcal{PR}$ contient $\mathbf{0}$, σ et $P_{i,p}$ pour tout $i \leq p \in \mathbb{N}$
- $\mathcal{PR}$ est stable pour $\circ$ et $\circlearrowleft$.

Exemples de fonctions primitives récursives

Montrer que les fonctions suivantes sont dans $\mathcal{PR}$.

- $f_1() = 1$
- $f_2(x) = 1$

cf le tableau interactif

- $f_3(x, y) = x + y$
- $f_4(x, y) = x * y$

Idée:
- pour l'addition, appliquer la récursion primitive pour ajouter 1 à x pendant y itérations,
- pour la multiplication, appliquer la récursion primitive pour ajouter x à x pendant y itérations.

Remarque, une fois que f1, f2, f3 et f4 sont dans PR, on peut les utiliser pour démontrer que d'autres fonctions sont dans PR. On est pas obligé de repartir des 3 fonctions initiales à chaque fois.

Fonctions récursives

Calculable $\neq$ Fonction primitive récursive

Il existe des fonctions non primitives récursives qui sont calculables.

- fonction d'Ackerman
- successeur de la diagonale de $\mathcal{PR}$ (cf le tableau interactif)

Fonctions récursives

Fonction de *Minimisation non bornée*

$\mu_p : \mathcal{F}_{p+1} \rightarrow \mathcal{F}_p$
Soit $x = (x_1, x_2, \ldots, x_p)$.

si f(x, m) est définie pour tout m <= n

$$\mu(f)(x) = \min(n | f(x, n) > 0) \text{ ~~si ce minimum existe~~}$$

On omettra p près du symbole μ sauf abiguïté.

ATTENTION : la fonction de minimisation peut donner naissance à des fonctions partielles ! (non définie sur tout N^p)

Comme expliqué dans les slides du tableau intéractif,
si f(x, m) n'est pas défini pour tout m <= n alors on peut créer une fonction mu(f) qui ne peut pas être calculable par une machine de Turing, ce qui est en contradiction avec le théorème de la slide 12.

ça ne change pas le fait que mu crée des fonctions partielles. Ce qui change c'est l'ensemble de définition de mu(f) qui devient plus restreint et donc plus simple à calculer.

Fonctions récursives

Fonctions μ-récursives : $\mathcal{R}$

L'ensemble $\mathcal{R}$ est l'ensemble des fonctions de $\mathcal{F}$ telles que :

- $\mathcal{R}$ contient $\mathcal{PR}$
- $\mathcal{R}$ est stable pour $\circ$, $\circlearrowleft$ et μ.

ATTENTION : puisque μ crée des fonctions partielles, il faut, en toute rigueur, redéfinir $\circ$, $\circlearrowleft$ pour des fonctions partielles.

Exemple de fonctions récursives

La fonction d'Ackerman $\mathcal{A} \in \mathcal{F}_2$ est récursive.

- $\mathcal{A}(0, m) = m + 1$
- $\mathcal{A}(m + 1, 0) = \mathcal{A}(m, 1)$
- $\mathcal{A}(m + 1, n + 1) = \mathcal{A}(m, \mathcal{A}(m + 1, n))$

Fonctions récursive VS Fonctions calculables

Théorème

$\mathcal{R}$ est l'ensemble des fonctions calculables par une machine de Turing.

Pour le démontrer :
- si f est une fonction récursive alors elle est calculable (cf les slides du panneau interactif)
- si f est une fonction calculable alors elle est récursive. Pour cela, il faut montrer qu'on peut simuler une machine de Turing avec une fonction récursive. La démonstration n'est pas triviale. On verra si on a le temps comment simuler une machine avec une formule booléenne, ce qui est une meilleur point de départ à mon sens pour voir comment on peut simuler une machine avec un objet mathématique qui n'a rien en commun avec une machine.

Machine de Turing universelle

Définition

Une machine de Turing $\mathcal{U}$ est dite *universelle* si elle peut simuler n'importe quelle autre machine.

- Toute machine de turing $\mathcal{M}$ peut se décrire à l'aide d'un mot $w_{\mathcal{M}}$ binaire.
- Avec le mot $w_{\mathcal{M}}Sx$ en entrée, $\mathcal{U}$ calcule le même mot que $\mathcal{M}$ calulerait avec x en entrée. Si $\mathcal{M}$ ne s'arrête pas alors $\mathcal{U}$ non plus.

D'après wikipédia, la plus petite machine de Turing universelle contiendrait 2 états et 18 symboles.
La plus petite avec 2 symboles contiendrait 15 états.
https://en.wikipedia.org/wiki/Universal_Turing_machine

Vous en trouverez une autre ici :
http://web.mit.edu/manoli/turing/www/turing.html

Fonctions non calculables

Fonction non calculable

Il existe des fonctions qui ne peuvent être calculées par une machine de Turing.

un argument simple est que le nombre de fonctions de N -> N n'est pas dénombrable, contrairement aux machines de Turing

Théorème

Une fonction $f \in \mathcal{F}_p$ est calculable si et seulement si le langage $L = \{(x_1, x_2, \ldots, x_p, f(x_1, x_2, \ldots, x_p))\}$ est décidable.

Corollaire

Il existe des langages indécidables.

Ce théorème montre que l'ensembles des fonctions de N^p dans N^q peut se restreindre à l'ensemble des fonctions de N dans {0, 1}. Donc tout résultat (positif ou négatif) sur ce simple ensemble de fonctions se reporte sur toutes les fonctions.

Exemples de fonctions non calculables

Les fonctions suivantes ne sont pas calculables :

- $MAXBB(n)$ =le plus grand nombre calculable par une machine de Turing avec n états.
- $BB(n)$ =le plus grand nombre de 1 que peut écrire une machine de Turing avec n états quand le mot en entrée est vide.
- $Tur(n)$ =la n^e décimal du nombre de Turing
  (cette décimale vaut 1 si la n-ieme machine de Turing (dans un ordre arbitraire) s'arrête et 0 sinon autrement dit, le problème de l'arrêt (cf slide suivante) peut être encodé dans un seul nombre (réel et infini), autrement dit le calcul d'un nombre bien défini est aussi difficile que le calcul d'une fonction)

Problème de l'arrêt

$L_{arrêt} = \{(M, x) \mid M \text{ s'arrête avec } x \text{ en entrée}\}$

Langage associé au problème de l'arrêt. Quand on dit que le problème est indécidable, il faut comprendre que le langage associé est indécidable.

Problème de l'arrêt

Soit $\mathcal{M}$ une machine et x un mot, est-ce que $\mathcal{M}$ s'arrête pour le mot x ?

Théorème

Le problème de l'arrêt est indécidable.

Autres exemples de langages indécidables

Les langages suivants sont indécidables :

- Peut-on libérer une variable x dans un programme juste après sa création ?
- La machine $\mathcal{M}$ s'arrête-t-elle pour toutes les entrées ?
- Pavage du quart de plan

Si je vous donne des carrés étiquetés et des règles qui permettent ou pas de poser des carrés côte à côte (horizontalement et verticalement), peut-on paver tout le quart de plan?

Pour montrer qu'un langage est indécidable, il faut montrer que décider un tel langage permettrait de résoudre le problème de l'arrêt (ou tout autre langage indécidable). On peut aussi essayer de se ramener à une contradiction comme pour le problème de l'arrêt.

Dans le cas du pavage de plan, il faut donc essayer d'encoder l'arrêt d'une machine de Turing dans un pavage de plan.

Langage acceptable

Définition

Un langage L est dit *acceptable* s'il existe une machine de Turing $\mathcal{M}$ qui s'arrête pour tout mot de L. On dit aussi *récursivement énumérable* ou *semi-décidable*.
On note RE l'ensemble des langages récursivement énumérables.-

Théorème

Un langage L décidable est acceptable.

Énumérer un langage

Définition

Soit $\mathcal{M}$ une machine de Turing avec 2 bandes, la machine $\mathcal{M}$ *énumère* le langage L si elle écrit sur la seconde bande chaque mot de L dans un ordre arbitraire.

Si le langage L est infini, $\mathcal{M}$ ne s'arrête pas.

Théorème

$L \in RE$ si et seulement s'il existe une machine qui énumère L.

⇐ soit M' qui utilise M jusqu'à ce que M énumère le mot x.
Si $x \in L$, M' s'arrête $\Rightarrow L$ acceptable.

⇒

Exemples

Montrer que les langages suivants sont énumérables.

1. $\{(10)^n, n \in \mathbb{N}\}$ — 01 concaténation n fois. Voir tableau interactif
2. $\{(\mathcal{M}, x)$ tels que $\mathcal{M}$ ne s'arrête pas avec x en entrée $\}$ — Impossible. C'était un piège
3. L'ensemble des théorèmes démontrables d'un modèle axiomatique.

On sait que $\{(\Pi, x)$ tq Π s'arrête avec $x\} \in$ ERÉ (on fait tourner $\Pi(x)$, si elle s'arrête, on s'arrête)

Donc si $\{(\Pi, x)$ tq Π s'arrête pas avec $x\} \in$ ERÉ, on peut décider si Π s'arrête sur x et répond au pb de l'arrêt.

En effet, on a une machine M1 qui s'arrête si M(x) s'arrête
et M2 qui s'arrête si M(x) ne s'arrête pas.
On fait tourner les 2 en parallèle et on regarde laquelle s'arrête en premier.

Plus généralement,
L est décidable ssi (L et son complémentaire sont acceptables)

Inacceptabilité

Théorème

Il existe des langages qui ne sont pas acceptables.

par exemple, le 2 e langage de la slide précédente.