

LINFO1123

CALCULABILITÉ, LOGIQUE ET COMPLEXITÉ

YVES DEVILLE

FORMULAIRE

Dylan GOFFINET

2021-2022

Table des matières

1	Concepts	1
1.1	Fonction	1
1.2	Enumérable	1
1.2.1	Diagonalisation de Cantor	1
2	Résultats fondamentaux	2
2.1	Fonction calculable	2
2.1.1	Ensemble récursif	2
2.1.2	Ensemble récursivement énumérable	2
2.1.3	Propriétés	2
2.1.4	Numérotation	2
2.2	Calculabilité	3
2.2.1	Problème de l'arrêt	3
2.2.2	Hoare-Allison	4
2.2.3	Rice	4
2.2.4	Paramétrisation	4
2.2.5	Point fixe	5
3	Modèles	6
3.1	ND-Java	6
3.2	ND-Récursif	6
3.2.1	ND-Récursif énumérable	6
3.2.2	Propriétés	6
4	Réductions	7
4.1	Réduction algorithmique (calculabilité)	7
4.1.1	Propriétés	7
4.2	Réduction fonctionnelle (complexité)	7
4.3	Réduction polynomiale	7
4.4	Classes de complexité	7
4.5	Relations entre classes de complexité	8
4.5.1	Déterministe VS non déterministe	8
4.5.2	Time VS Space	8
4.6	Formalismes de calculabilité	8
4.6.1	Caractéristiques de formalisme	8
4.6.2	Propriétés	8
5	Preuve supplémentaire	9
5.1	L'ensemble des fonctions totales n'est pas énumérable	9

1 | Concepts

1.1 Fonction

Soit $f : A \rightarrow B$:

- **Domaine** de f : $\text{dom}(f) = \{a \in A \mid f(a) \neq \perp\}$
- **Image** de f : $\text{im}(f) = \{b \in B \mid \exists a \in A : b = f(a)\}$
- f est fonction **totale** ssi $\text{dom}(f) = A$ ($\nexists a \in A : f(a) = \perp$)
- f est fonction **partielle** ssi $\text{dom}(f) \subseteq A$ (f totale est partielle mais f partielle n'est pas forcément totale)
- f est **surjective** ssi $\text{im}(f) = B$ (tout y a au moins un x)
- f est **injective** ssi $\forall a, a' \in A : a \neq a' \rightarrow f(a) \neq f(a')$ (tout x a un y différent)
- f est **bijjective** ssi f est totale, injective et surjective (tout y a un et un seul x)

1.2 Enumérable

Un ensemble est énumérable s'il est soit fini ou s'il a le même cardinal que \mathbb{N}^1 (si on peut le mettre en bijection avec \mathbb{N}). En informatique, un programme est une chaîne finie de caractères \rightarrow énumérable.

1.2.1 Diagonalisation de Cantor

Soit $E = \{x \in \mathbb{R} \mid 0 < x \leq 1\}$. **E est non énumérable**²

Preuve :

1. Supposons E énumérable. Il existe donc une énumération des éléments de $E : x_0, \dots, x_k, \dots$

	1 digit	2 digit	3 digit	...	$k + 1$ digit	...
x_0	x_{00}	x_{01}	x_{02}	...	x_{0k}	...
x_1	x_{10}	x_{11}	x_{12}	...	x_{1k}	...
x_2	x_{20}	x_{21}	x_{22}	...	x_{2k}	...
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	...
x_k	x_{k0}	x_{k1}	x_{k2}	...	x_{kk}	...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots

FIGURE 1.1: Construire une table t.q. le nombre $x_k = 0, x_{k0}x_{k1}x_{k2} \dots x_{kk} \dots$

2. Prendre la diagonale ($d = 0, x_{00}x_{11}x_{22} \dots x_{kk} \dots$)
3. Modifier la diagonale t.q.

$$x'_{ii} = \begin{cases} 5 & \text{si } x_{ii} \neq 5 \\ 6 & \text{sinon} \end{cases}$$

Ce qui donne $d' = 0, x'_{00}x'_{11}x'_{22} \dots x'_{kk} \dots$ ($d' \in E$)

4. Contradiction : Comme E est énumérable, et que $d' \in E$, alors d' doit être dans l'énumération. Or, si $d' = x_p$, on a :

$$\begin{aligned} d' &= 0, x_{p0}x_{p1}x_{p2} \dots x_{pp} \dots \\ &= 0, x'_{p0}x'_{p1}x'_{p2} \dots x'_{pp} \dots \end{aligned}$$

5. Conclusion : E n'est pas énumérable

1. L'ensemble \mathbb{N} est l'ensemble des entiers positifs $\mathbb{N} = \{0, 1, 2, 3, \dots\}$

2. Il n'existe pas de bijection entre E et \mathbb{N} c.à.d. qu'il y a beaucoup plus de réels entre 0 et 1 qu'il n'y a d'entiers positifs

2 | Résultats fondamentaux

2.1 Fonction calculable

Une fonction $f : \mathbb{N} \rightarrow \mathbb{N}^1$ est calculable ssi il existe un programme qui, recevant comme données n'importe quel nombre naturel x , fourni comme résultat $f(x)$ s'il est défini, sinon \perp (s'il ne se termine pas ou erreur).

2.1.1 Ensemble récursif

On dit que l'ensemble A est récursif ssi il existe un programme qui prend en input x et qui renvoi (c.à.d un ensemble récursif est un ensemble pour lequel on est capable de dire si un élément y appartient) :

$$\begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases}$$

Le programme calcule donc une fonction totale. Exemple : $\{x \in \mathbb{N} \mid x \text{ pair}\}$.

2.1.2 Ensemble récursivement énumérable

On dit que l'ensemble A est récursivement énumérable ssi il existe un programme qui prend en input x et qui renvoi (tôt ou tard) :

$$\begin{cases} 1 & \text{si } x \in A \\ \text{Un autre résultat, ou ne se termine pas} & \text{si } x \notin A \end{cases}$$

2.1.3 Propriétés

- A récursif $\Rightarrow A$ récursivement énumérable
- A récursif $\Leftrightarrow \bar{A}$ récursif²
- A récursivement énumérable et \bar{A} récursivement énumérable $\Leftrightarrow A$ récursif
- A fini ou \bar{A} fini $\Rightarrow A$ et \bar{A} récursif

2.1.4 Numérotation

Soit P l'ensemble des programmes syntaxiquement corrects.

- P est énumérable récursif
- $P = P_0, P_1, \dots$ (sans répétition)
- P_k est le programme numéro k dans P
- φ_k est la fonction numéro k calculée par le programme P_k ($\varphi : \mathbb{N} \rightarrow \mathbb{N}$)

1. Se lit : "une fonction de \mathbb{N} dans \mathbb{N} ". L'ensemble $\mathbb{N} \rightarrow \mathbb{N}$ est non énumérable (l'ensemble des problèmes a un plus grand cardinal que l'ensemble des programmes), voir 5.1

2. \bar{A} est le complément de A

2.2 Calculabilité

2.2.1 Problème de l'arrêt

Soit la fonction $halt : P \times \mathbb{N} \rightarrow \mathbb{N}$ t.q. $halt(n, x) = \begin{cases} 1 & \text{si } \varphi_n(x) \neq \perp \\ 0 & \text{sinon} \end{cases}$

halt n'est pas calculable.

Preuve

Supposons *halt* calculable.

1. Construire la table

	0	1	2	...	k	...
P_0	$halt(0, 0)$	$halt(0, 1)$	$halt(0, 2)$...	$halt(0, k)$...
P_1	$halt(1, 0)$	$halt(1, 1)$	$halt(1, 2)$...	$halt(1, k)$...
P_2	$halt(2, 0)$	$halt(2, 1)$	$halt(2, 2)$...	$halt(2, k)$...
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	...
P_k	$halt(k, 0)$	$halt(k, 1)$	$halt(k, 2)$...	$halt(k, k)$...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots

FIGURE 2.1: Table des valeurs de la fonction *halt*

2. Prendre la diagonale : $d(n) = halt(n, n)$
3. Modifier la diagonale :

$$d'(n) = \begin{cases} 1 & \text{si } halt(n, n) = 0 \\ \perp & \text{si } halt(n, n) = 1 \end{cases}$$

Si *halt* est calculable, alors d' est calculable. Soit P_d le programme qui calcule cette fonction.

4. Contradiction :

$$d'(d) = \begin{cases} 1 & \rightarrow halt(d, d) = 0 \rightarrow P_d \text{ ne se termine pas OR } d'(d) = 1 \\ \perp & \rightarrow halt(d, d) = 1 \rightarrow P_d \text{ se termine OR } d'(d) = \perp \end{cases}$$

5. Conclusion : d' n'est pas calculable, donc *halt* n'est pas calculable.

2.2.2 Hoare-Allison

Soit un langage Q qui a des programmes Q_k et qui ne calcule que des fonctions totales :

- La fonction φ'_k est calculée par le programme Q_k
- L'interpréteur $interpret(n, x)$ de ce langage Q est calculable
- La fonction $halt(n, x)$ pour ce langage Q est calculable (fonction constante qui vaut 1)
- $interpret(n, x)$ n'est pas calculable dans Q

Preuve

Supposons $interpret$ calculable dans Q .

1. Construire la table

	0	...	k	...
Q_0	$interpret(0, 0)$...	$interpret(0, k)$...
\vdots	\vdots	\ddots	\vdots	...
Q_k	$interpret(k, 0)$...	$interpret(k, k)$...
\vdots	\vdots	\vdots	\vdots	\ddots

FIGURE 2.2: Table des valeurs de la fonction $interpret$

2. Prendre la diagonale : $d(n) = interpret(n, n)$
3. Modifier la diagonale : $d'(n) = interpret(n, n) + 1$ (calculable dans Q si $interpret$ calculable dans Q)
4. Contradiction : $d'(d) = interpret(d, d) + 1$ **OR** $d'(d) = \varphi'(d) = interpret(d, d)$
5. Conclusion : l'interpréteur de Q n'est pas calculable dans Q

2.2.3 Rice

Deux formulations :

1. Soit $A \subseteq \mathbb{N}$, si A récursif, $A \neq \emptyset$ et $A \neq \mathbb{N}$ alors $\exists i \in A$ et $\exists j \in \bar{A}$ t.q. $\varphi_i = \varphi_j$
2. Si $\forall i \in A$ et $\forall j \in \bar{A} : \varphi_i \neq \varphi_j$ alors A non récursif ou $A = \emptyset$ ou $A = \mathbb{N}$

→ Aucun programme ne peut dire si une fonction respecte des spécifications

Si $A \neq \emptyset$ et $A \neq \mathbb{N}$, $\forall i \in A, \forall j \in \bar{A} : \varphi_i \neq \varphi_j$. Alors A non récursif.

Preuve

1. On suppose A récursif.
On pose $P_k(x) \equiv \text{while}(\text{True})$ (c.à.d. $\varphi_k = \perp$).
Si $k \in \bar{A}$, comme $A \neq \emptyset \Rightarrow \exists m \in A$ et $\varphi_k \neq \varphi_m$

2. Construire $halt$:

$$halt(n, x) \equiv \begin{cases} \text{Construire le programme (sans l'exécuter) } P(z) \equiv P_n(x); P_m(z) \\ d = \text{numéro du programme } P(z) \\ \text{if } d \in A \text{ then } print(1) \text{ Si } P_n \text{ se termine, } \varphi_d = \varphi_m \rightarrow d \in A \\ \text{else } print(0) \text{ Si } P_n \text{ ne se termine pas, } \varphi_d = \varphi_k \rightarrow d \in \bar{A} \end{cases}$$

3. $halt$ n'est pas calculable, donc A est non récursif

2.2.4 Paramétrisation

Si un programme $P(a, b)$ existe, alors il existe un programme $P'_b(a)$ (où b est fixé) t.q. $P'_b(a) \equiv Exec P(a, b)$

Forme S-1-1

Il existe une fonction totale calculable $S_1^1 : \mathbb{N}^2 \rightarrow \mathbb{N}$ t.q. $\forall k : \varphi_k(x_1, x_2) = \varphi_{S_1^1(k, x_2)}(x_1)$

Forme S-m-n

$\forall m, n \geq 0, \exists$ une fonction totale calculable $S_n^m : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$ t.q. $\forall k : \varphi_k(x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}) = \varphi_{S_1^1(k, x_{n+1}, \dots, x_{n+m})}(x_1, \dots, x_n)$

2.2.5 Point fixe

Soit f une fonction totale calculable. Il existe k t.q. $\varphi_k = \varphi_f(k)$.

Preuve

On pose :

$$h(u, x) = \begin{cases} \varphi_{\varphi_u(u)}(x) & \text{si } \varphi_u(u) \neq \perp \\ \perp & \text{sinon} \end{cases} \quad (2.1)$$

Où $h(u, x)$ est calculable

$$h(u, x) = \varphi_{S(u)}(x) \quad (2.2)$$

Par application de $S - 1 - 1$

$$g(x) = f(S(x)) \quad (2.3)$$

Où g est totale calculable (car f et S le sont),

et f est donné par $k' : \varphi_{k'}(x) = g(x) = f(S(x))$.

On a que k' est une constante par l'équation 2.2 :

$$h(k', x) = \varphi_{S(k')}(x)$$

Par l'équation 2.1 et comme $g = \varphi_{k'}$:

$$h(k', x) = \varphi_{k'(k')}(x)$$

Par l'équation 2.3, on a que $\varphi_{k'} = g(x) = f(S(x))$, donc :

$$h(k', x) = \varphi_{f(S(x))}(x)$$

Si on pose que $S(k') = k$, on obtient :

$$\varphi_k(x) = \varphi_{f(k)}(x)$$

3 | Modèles

3.1 ND-Java

C'est un sous-ensemble de Java ("Non-Deterministic Java") On y ajoute fonction *choose*(*n*) renvoyant un entier aléatoire entre 0 et *n*. Cette fonction est non-déterministe car à un même input elle ne renvoie pas toujours le même output.

3.2 ND-Récuratif

A est ND-Récuratif si \exists un programme ND-Java t.q. s'il reçoit un input $x \in \mathbb{N}$:

- $x \in A$ alors \exists une exécution qui retourne 1
- $x \notin A$ alors pour toute exécution le résultat est 0

3.2.1 ND-Récuratif énumérable

Comme ND-Récuratif sauf que le cas $x \notin A$ ne se fini pas forcément.

3.2.2 Propriétés

- Récuratif \Rightarrow ND-Récuratif
- Récuratif énumérable \Rightarrow ND-Récuratif énumérable

4 | Réductions

4.1 Réduction algorithmique (calculabilité)

Un ensemble A est **algorithmiquement réductible** à un ensemble B ($A \leq_a B$) si en supposant B récursif, A est récursif.

Exemple :

Soit $P = \{n \mid \varphi_n \text{ renvoi un nombre pair}\}$

$HALT \leq_a P$ (si P énumérable, $HALT$ énumérable)

4.1.1 Propriétés

- Si $A \leq_a B$ et B récursif, alors A récursif
- Si $A \leq_a B$ et A non récursif, alors B non récursif
- $A \leq_a \bar{A}$
- $A \leq_a B \Leftrightarrow \bar{A} \leq_a B$
- Si A récursif, alors pour tout B , $A \leq_a B$
- Si $A \leq_a B$ et B récursivement énumérable, alors A pas nécessairement énumérable

4.2 Réduction fonctionnelle (complexité)

Un ensemble A est **fonctionnellement réductible** à un ensemble B ($A \leq_r B$) ssi il existe une fonction totale calculable f t.q. :

$$a \in A \Leftrightarrow f(a) \in B$$

4.3 Réduction polynomiale

Un ensemble A est **polynomialement réductible** à un ensemble B ($A \leq_p B$) ssi il existe une fonction totale calculable f de complexité temporelle polynomiale t.q. :

$$a \in A \Leftrightarrow f(a) \in B$$

Si $A \leq_p B$ et $B \in P$ alors $A \in P$.

4.4 Classes de complexité

- $DTIME(f)$: Ensemble récursif décidé par un programme en complexité temporelle $\mathcal{O}(f)$
- $DSPACE(f)$: Ensemble récursif décidé par un programme en complexité spatiale $\mathcal{O}(f)$
- $NTIME(f)$: Ensemble ND-récursif décidé par un programme en complexité temporelle $\mathcal{O}(f)$ (sur toutes les branches)
- $NSPACE(f)$: Ensemble ND-récursif décidé par un programme en complexité spatiale $\mathcal{O}(f)$ (sur toutes les branches)
- Classe P (Polynomiale) : $P = \bigcup_{i \geq 0} DTIME(n^i)$
- Classe NP (Non-Polynomiale) : $P = \bigcup_{i \geq 0} NTIME(n^i)$

4.5 Relations entre classes de complexité

4.5.1 Déterministe VS non déterministe

- Si $A \in NTIME(f)$ alors $A \in DTIME(c^f)$
- Si $A \in NSPACE(f)$ alors $A \in DSPACE(f^2)$
- $NPSPACE(f) = DSPACE(f)$

4.5.2 Time VS Space

- Si $A \in NTIME(f)$ alors $A \in NSPACE(f)$
- Si $A \in DTIME(f)$ alors $A \in DSPACE(f)$
- Si $A \in NSPACE(f)$ alors $A \in NTIME(c^f)$
- Si $A \in DSPACE(f)$ alors $A \in DTIME(c^f)$

4.6 Formalismes de calculabilité

Soit D un nouveau formalisme de calculabilité :

4.6.1 Caractéristiques de formalisme

- SD (Soudness des Descriptions) : toute fonction D -calculable est calculable
- CD (Complétude des Définitions) : toute fonction calculable est D -calculable
- SA (Soudness Algorithmique) : l'interpréteur de D est calculable
- CA (Complétude Algorithmique) : si $p \in L$ (L est par exemple Java), que $p' \in D$ et que $p \equiv p'$ (calculent la même fonction), alors équivalence des formalismes
- U (Description Universelle) : l'interpréteur de D est D -calculable
- S ($S - m - n$ Affaiblie) : $\forall d \in S \exists S : d(x, y) = [S(x)](y)$

4.6.2 Propriétés

- $SA \Rightarrow SD$
- $CA \Rightarrow CD$
- SD et $U \Rightarrow SA$
- CD et $S \Rightarrow CA$
- SA et $CD \Rightarrow U$
- CA et $SD \Rightarrow S$
- S et $U \Rightarrow S - m - n$
- SA et $CA \Leftrightarrow SD$ et CD et U et S
- SA et CD et $S \Leftrightarrow CA$ et SD et U

5 | Preuve supplémentaire

5.1 L'ensemble des fonctions totales n'est pas énumérable

Soit F l'ensemble des fonctions totales telles que $f : \mathbb{N} \rightarrow \mathbb{N}$.

F est non énumérable.

Preuve

Supposons F énumérable. Il existe donc une énumération des éléments de $F : f_0(0), f_1(0), \dots$

1. Construire la table

	1	1	2	...	k	...
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$...	$f_0(k)$...
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$...	$f_1(k)$...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$...	$f_2(k)$...
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	...
f_k	$f_k(0)$	$f_k(1)$	$f_k(2)$...	$f_k(k)$...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots

FIGURE 5.1: Table des résultats de la fonction f

2. Prendre la diagonale d qui est aussi une fonction de $\mathbb{N} \rightarrow \mathbb{N}$ ($d \in F$)
3. Modifier la diagonale pour obtenir d' t.q. :

$$f'_i(j) = \begin{cases} 5 & \text{si } f_i(j) \neq 5 \\ 6 & \text{sinon} \end{cases}$$

Où $f_i(j)$ est le résultat de la fonction f avec le numéro i pour la donnée j .

4. Contradiction :

Comme F est énumérable et que $d' \in F$, alors d' doit être dans l'énumération. Or si d' a le numéro p on a :

$$\begin{aligned} d' &= f_p(0), f_p(1), f_p(2), \dots, f_p(p), \dots \\ &= f'_p(0), f'_p(1), f'_p(2), \dots, f'_p(p), \dots \end{aligned}$$

Si $f_p(0)$ vaut 5, on a $(f_p(0) = 5) \neq (f'_p(0) = 6)$.

5. Conclusion :

F n'est pas énumérable.