L2 ISTN notes personelles

Alex Videcoq

Calculabilitée 3ème semestre

vPre 0.0.1

ISTIC

Responsable de CAL Prof. Gilles Lesventes
Cursus L2 ISTN

Rennes 2025
(Draft - February 17, 2025)

Contents

Introduction

La calculabilité est un domaine fondamental de l'informatique théorique qui explore les limites de ce que les machines peuvent ou ne peuvent pas calculer. Ce document est une compilation complète de notes et de concepts pour le cours intitulé *Calculabilité*, dispensé dans le cadre de la Licence 2 mention Informatique. Ce cours vise à fournir aux étudiants une compréhension approfondie des concepts clés de la calculabilité et de la complexité, en les dotant des compétences nécessaires pour aborder des problèmes fondamentaux de l'informatique théorique.

Le cours aborde tout d'abord les concepts fondamentaux de la **calculabilité** **Chapter 1** → p.1, y compris les théorèmes de Cantor, le problème de l'arrêt, la réduction calculatoire et le théorème de Rice. Ces notions permettent aux étudiants de comprendre les limites intrinsèques des algorithmes et de reconnaître les problèmes qui ne peuvent être résolus de manière algorithmique.

Ensuite, le cours explore la **sémantique des langages de programmation** **Chapter 2** → p.7, en s'attardant sur la syntaxe et la sémantique du langage WHILE, la définition d'un interpréteur et la réduction à un sous-langage, le langage FOR. Les étudiants apprennent à définir formellement les comportements des programmes et à explorer les implications de la thése de Church.

Enfin, une attention particulière est accordée à la **complexité des algorithmes** **Chapter 3** → p.8, où les concepts de coût asymptotique, de classes de complexité P et NP, et le théorème de Cook sont introduits. Ce module permet aux étudiants de mieux comprendre l'efficacité des algorithmes et les difficultés liées à la résolution de problèmes NP-complets.

Résumé Rapide

Calculabilité Chapter 1 → p.1 : Traite des théorèmes de Cantor, du problème de l'arrêt, de la réduction calculatoire et du théorème de Rice.

Sémantique des langages de programmation Chapter 2 → p.7 : Analyse la syntaxe et la sémantique du langage WHILE, la mise en œuvre d'un interpréteur, et l'étude du langage FOR.

Complexité des algorithmes Chapter 3 → p.8 : Explique les concepts de complexité asymptotique, les classes de complexité P et NP, et le théorème de Cook.

1 Introduction à la Calculabilité

La *calculabilité* est un domaine de la logique mathématique et de l'informatique théorique qui vise à identifier les limites de ce qui peut être calculé par un algorithme. Cette discipline ne se préoccupe pas de l'efficacité des algorithmes, qui est l'objet de la théorie de la complexité, mais seulement de l'existence ou de la non-existence d'un algorithme qui résout un problème donné, sur un ordinateur idéalisé tel qu'une machine de Turing, démontrée équivalente en possibilités à tous les ordinateurs existants. La notion la plus centrale de la calculabilité est celle de fonction calculable. Son intérêt est justifié par la *thèse de Church* [1], qui affirme que les fonctions calculables avec la définition formelle correspondent exactement aux fonctions qui peuvent être calculées par un humain ou une machine, dans le monde physique. Cependant, la calculabilité ne se limite pas à séparer les fonctions calculables des fonctions non calculables, mais cherche aussi à comparer les fonctions non calculables entre elles, en s'appuyant sur la notion de réduction pour affirmer (informellement) que certaines fonctions sont plus "incalculables" que d'autres. Les niveaux d'incalculabilité sont appelés *degrés de Turing*, et une partie de la calculabilité consiste à étudier leur structure.

1.1 Notions fondamentales de la calculabilité

Programmes

Definition 1.1. Programme :
Un programme est un texte *fini*.
On notera $\mathbb{P}$ l'ensemble des programmes. C'est un ensemble fini.

> **Note.** Précision :
> Ici aucune hypothèse n'est faite sur le jeu de charactères, le lexique ou le langage de programmation.

Fonctions

Definition 1.2. Fonction :
Soient $\mathbb{D}$ et $\mathbb{D}'$ deux ensembles de valeurs, une *fonctions* $f : \mathbb{D} \to \mathbb{D}'$ est un ensemble de paires $\langle v, v' \rangle$ telles que $v \in \mathbb{D}$ et $v' \in \mathbb{D}'$, et vérifie :

$$\forall v \in \mathbb{D}, \forall v', v'' \in \mathbb{D}', [\langle v, v' \rangle \in f \wedge \langle v, v'' \rangle \in f \Rightarrow v' = v''] \tag{1.3}$$

On appelle v une *entrée* (en pratique, ce serait un *paramètre*), et v' une *sortie* ou *l'image de v par f*.
$\mathbb{D}$ est appelé son *domaine* et $\mathbb{D}'$ son *co-domaine*.

L'ensemble des valeurs v telles que $\langle v, v' \rangle$ appartient à une fonction f est le *domaine de définition* de f : $\{v \in \mathbb{D} | \exists v'. \langle v, v' \rangle \in f\}$.
Il peut être infini.
Si le domaine de définition de f est égal au domaine $\mathbb{D}$ on dit que la fonction f est *totale*. Elle est définie partout, $\forall v \in \mathbb{D}. \exists v'. \langle v, v' \rangle \in f$.
Sinon, on dit qu'elle est *partielle*, il existe des points ou elle n'est pas définie. Si x est un de ces point, faire référence à $f(x)$ n'à aucun sens.

Il existe deux points de vue pour décrire une fonction :

- **Point de vue extensionnel :** On considère qu'une fonction est entièrement définie par son *graphe* (l'ensemble des couples (x, y) qui la constitue) :

  Example 1.4.

  $$d = \{(0,0), (1,2), (2,4), (3,6), \dots\} \tag{1.5}$$

  ⌟

  Le point de vue *extensionnel* sur les fonctions les résume à leur comportement *d'entrée-sortie*.

- **Point de vue intensionnel :** Ici, on considère une fonction comme un procédé de calcul défini par un enchaînement d'opérations.

  Example 1.6.

  $$f : x \mapsto x + x \tag{1.7}$$
  $$g : x \mapsto 2x \tag{1.8}$$

  Ici, les définitions des fonctions f et g dénotent différentes fonctions dont on pourrait montrer qu'elles sont extensionnellement équivalentes parce qu'elles produisent partout le même effet.
  Ce principe est à rapprocher de la définition d'un ensemble en compréhension d'un ensemble :

  $$d = \{(x, y) | y = x + x\} \tag{1.9}$$

  ⌟

Problèmes de decision

Certaines *fonctions* sont ce que l'on appelle des *problèmes de décision* (voir Definition 1.10 →p.2). Elles définissent si une propriété est *vraie* en rendant les valeurs *vrai* ou *faux*.
L'ensemble $\{vrai, faux\}$ est noté $\mathbb{B}$.

Todo. Complément.
Mettre en évidence le lien entre les problèmes de décision et l'algèbre de Boole.

Definition 1.10. Problème de décision :
Un problème de décision est une fonction f totale de $\mathbb{D}$ dans $\mathbb{B}$. On écrit

$f : \mathbb{D} \to \mathbb{B}$.

Une *instance* d'un problème de décision, une donnée de $\mathbb{D}$ pour laquelle on veut résoudre le problème. On dit qu'une instance est *positive* si son image est `vrai` et *négative* si son image est `fausse`. L'ensemble des problèmes de décision est *beaucoup plus gros* que celui des programmes.

Sémantique

Definition 1.11. Sémantique d'un programme :
La sémantique des programmes est modélisé par une fonction $\mathcal{SEM}$ définie des programmes $\mathbb{P}$ vers les fonctions $\mathbb{D} \to \mathbb{D}$. On écrit $\mathcal{SEM} : \mathbb{P} \to \mathbb{D} \to \mathbb{D}$ et $\mathcal{SEM}(\mathtt{p})$ la sémantique du programme `p`.

Puisque que la sémantique d'un programme est une fonction, $\mathcal{SEM}(\mathtt{p})$ est une fonction qui demande à être appliquée à des données. On écrira donc $\mathcal{SEM}(\mathtt{p})(v)$ pour designer le résultat de l'application de la sémantique d'un programme `p` a un programme v.

Note. Différencier la sémantique du programme.
Il est tentant d'écrire $\mathtt{p}(v)$, mais `p` n'est qu'un texte, il ne peut rien tout seul.

La sémantique d'un programme est donc un ensemble de paires $\langle valeur, image \rangle$.

Quand l'exécution d'un programme ne répond rien pour certaines valeurs d'entrée, boucle, s'arrête brutalement sur une condition d'erreur (ex. division par 0), la fonction $\mathcal{SEM}$ n'est pas définie sur ces valeurs.

Note. Définir le non défini.
On ne peut pas augmenter le co-domaine d'une nouvelle valeur, `non-def`, qui servirait à représenter qu'une fonction n'est pas définie en un point v, tel que $\langle v, \mathtt{non\text{-}def} \rangle \in \mathcal{SEM}(\mathtt{p})$ plutôt que $\neg\exists w. \langle v, w \rangle \in \mathcal{SEM}(\mathtt{p})$.
Cela reviendrait à définir le non défini, cela reviendrait à dire que toutes les anomalies sont prévues, or c'est impossible.

D'autre part, la théorie de la calculabilité montre qu'il n'existe pas de modèle de calcul qui ne réalise que des fonctions totales, et les réalise toutes. On s'en tiendra donc à l'idée que des paires peuvent manquer dans la définition d'une fonction.

Il existe aussi des programmes mal formes, qui seront refusés par un compilateur par exemple, du point de vue de la sémantique un programme mal forme n'est pas un programme.

Calculabilité et décidabilité

Definition 1.12. Fonctions calculables.
Lorsqu'une fonction totale est la sémantique d'un programme, on dit qu'elle est *calculable*. On dit sinon qu'elle est *non calculable*.

Definition 1.13. Problèmes décidables.
Lorsqu'un problème de décision est calculable, on dit qu'il est *décidable*. On dit sinon qu'il est *indécidable* ou non décidable.

Result 1.14. La fonction qui teste l'arrêt d'un programme sur une donnée, tous deux passés en paramètres, n'est pas calculable. [2] ⌟

Definition 1.15. Pour chaque valeur, $v \in \mathbb{D}$ l'exécution d'un programme $\mathtt{p}$ peut s'arrêter, $\exists w.\langle v, w\rangle \in \mathcal{SEM}(\mathtt{p})$, ou boucler $\neg\exists w.\langle v, w\rangle \in \mathcal{SEM}(\mathtt{p})$.
Le problème de l'arrêt consiste à réaliser une fonction Arrêt : $\mathbb{P} \times \mathbb{D} \to \mathbb{B}$ telle que $\mathtt{p}, v \mapsto \exists w.\langle v, w\rangle \in \mathcal{SEM}(\mathtt{p})$

Result 1.16. Les fonctions qui prennent des programmes en paramètre et dont le résultat dépend uniquement de leur sémantique ne sont jamais calculables. C'est le théorème de Rice. [3] ⌟

Note. Sur le théorème de Rice.
Le théorème de Rice ne pose de limites que sur l'analyse statique du programme, c'est-à-dire sur tous les problèmes qui peuvent être déduits sans lancer le programme. Il ne rentre donc pas en conflit avec les fonctions de compilation, même si les compilateurs actuels sont de plus en plus puissants et performants, leur analyse du code reste statique.
Lectures conseillées :

- Sémantiques mécanisées, Xavier Leroy, Collège de France
- Data flow analysis, Thomas Jensen, Université de Rennes

La thèse de Church

Definition 1.17. Modèle de calcul
Un modèle de calcul est la définition formelle d'une règle de calcul qui compose des opérations élémentaires ne recourant chacune qu'à des moyens finis.

Result 1.18. La notion de calculabilité ne dépend pas du modèle de calcul. ⌟

De nombreuses propositions de langages de programmation et de sémantiques de programmes ont été faites. À chaque fois, les mêmes capacités de calcul ont été trouvées. Autrement dit, à chaque fois, les mêmes fonctions étaient non calculables. Cette observation se formalise dans une conjecture, appelée la *Thèse de Church* [1], énoncée par Alonzo Church et ses collègues à partir de 1936.
On dit des modèles de calcul qui permettent de programmer tout ce qui est

calculable qu'ils ont la puissance du calculable, ou qu'ils sont équivalents à une machine de *Turing universelle*, du nom de l'un des premiers modèles de la calculabilité, ou encore qu'ils sont calculatoirement complets. La plupart des langages de programmation ont la puissance du calculable.

Programmes WHILE

Les programmes WHILE dont la principale caractéristique est d'offrir une boucle `while c do i od` qui commande d'exécuter l'instruction `i` tant que la condition `c` est vraie. La condition `c` peut cesser d'être vraie par l'effet de l'exécution de l'instruction `i`. Puisque l'exécution de ces programmes peut boucler, leur sémantique peut consister en une fonction partielle. En contrepartie, ils ont la puissance du calculable.

Programmes FOR

Le modèle des programmes FOR, en remplace la boucle `while c do i od` par une boucle `for n do i od` qui commande d'exécuter `n` fois l'instruction `i`. Ce nombre d'itérations `n` est calculé avant d'exécuter la boucle.

Result 1.19. Il n'est pas possible de réaliser toutes les fonctions calculables dans un modèle de calcul où l'exécution des programmes termine toujours. ⌟

Fonction d'Ackermann

Le moyen le plus direct de montrer qu'il existe des fonctions calculables non programmables dans le langage FOR est d'en exhiber un exemple concret tel que la fonction d'Ackermann :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases} \tag{1.20}$$

Complexité des fonctions

Result 1.21. Le coût minimal pour les réaliser par programme est une propriété intrinsèque des fonctions. Dans certains cas, ce coût est énorme et rend la fonction irréalisable en pratique. On utilise d'ailleurs le terme *impraticable*. ⌟

Pour certaines fonctions, on pourra établir un coût de calcul *minimum intrinsèque*. Pour d'autres, on ne connaît qu'un coût de calcul *minimum empirique* : celui du programme le plus efficace à ce jour qui les réalise.

$\mathcal{P}$ vs. $\mathcal{NP}$

Definition 1.22. Les fonctions $\mathcal{P}$
On appelle $\mathcal{P}$ l'ensemble des fonctions dont le coût du calcul en fonction de la taille des entrées est majoré par une fonction polynomiale de la taille de l'entrée.

Definition 1.23. Les fonctions $\mathcal{NP}$
On appelle $\mathcal{NP}$ l'ensemble des fonctions dont le coût de la vériffication des réponses est majoré par une fonction polynomiale.

Result 1.24. Les ensembles P et N P ont été déffinis vers 1970 ; ils s'intègrent dans une classification des fonctions qui est riche, mais des questions restent sans réponse à leur sujet.
On ne sait pas si $\mathcal{P} \neq \mathcal{NP}$, cela constitue l'un des problèmes ouverts les plus importants de notre époque, en mathématiques et en informatique. ⌟

La machine de Turing

Cette machine de papier a été déffinie en 1936 par Alan Turing et a servi de modèle théorique à la création des calculateurs tels que nous les connaissons actuellement. Les notions aussi fondamentales que celles de l'indécidabilité ou de l'interprétation d'un programme au moyen d'un autre programme ont été déffinies par Alan Turing lui-même, dès 1936. De ce modèle ont découlé les déffinitions d'autres modèles basés sur des automates.

1.2 Théorèmes de Cantor

1.3 Le problème de l'arrêt

1.4 La réduction calculatoire

1.5 Le théorème de Rice

2 Sémantique des langages de programmation

2.1 Introduction à la syntaxe et à la sémantique

2.2 Le langage WHILE

Définition syntaxique

Définition sémantique formelle

Programmation dans WHILE

2.3 Sous-langage FOR

2.4 La thése de Church

2.5 Projets pratiques

Implémentation d'un interpréteur

Fragment de compilateur

3 Complexité des algorithmes

3.1 Définition intuitive de la complexité

3.2 Coût asymptotique et notations

3.3 Classes de complexité

Problèmes P et NP

Réduction polynomiale

3.4 Problèmes NP-complets

3.5 Le théorème de Cook

4 Travaux dirigés et travaux pratiques

4.1 Travaux dirigés

TD 1 - Nombres remarquables et ensembles de Cantor

Exercice 1 - Dénombrabilité de $\mathbb{Z}$ et $\mathbb{Q}$

Nous avons vu en cours que l'ensemble des rationnels positifs, $\mathbb{Q}^+$ était en bijection avec l'ensemble des entiers naturels $\mathbb{N}$. On dit alors que l'ensemble des nombres rationnels positifs est dénombrable. La cardinalité de l'ensemble des nombres naturels $\mathbb{N}$ se note $\aleph_0$ et on écrit $\|\mathbb{Q}^+\| = \|\mathbb{N}\| = \aleph_0$.

1. Montrer qu'il existe toujours une injection d'une union de deux ensembles dénombrables vers $\mathbb{N}$. En déduire qu'une union de deux ensembles dénombrables est dénombrable. Citer le théorème utilisé.

Soient A et B deux ensembles dénombrables.
Par définition, il existe des bijections :

$$f : A \to \mathbb{N} \quad \text{et} \quad g : B \to \mathbb{N} \tag{4.1}$$

On définit une fonction injective de $A \cup B$ vers $\mathbb{N}$ en intercalant des éléments de A et B :

$$h(x) = \begin{cases} 2f(x), & \text{si } x \in A \backslash B \\ 2g(x) + 1, & \text{si } x \in B \end{cases} \tag{4.2}$$

Cette fonction est injective, car les images de A sont des nombres pairs et celles de B sont des nombres impairs. Ainsi, chaque élément de $A \cup B$ a une image distincte dans $\mathbb{N}$. Réciproquement f^{-1} définit une injection de $A \cup B$ dans $\mathbb{N}$.

Theorem 4.3. (Cantor-Schröder-Bernstein, 1895).
Soit E et F deux ensembles. S'il existe une injection de E dans F et une injection de F dans E, E et F sont en bijection.

2. En déduire que l'ensemble des entiers relatifs $\mathbb{Z}$ et l'ensemble des rationnels $\mathbb{Q}$ sont dénombrables :
$\|\mathbb{Z}\| = \|\mathbb{Q}\| = \|\mathbb{N}\| = \aleph_0$

Definition 4.4. $\mathbb{Z}$.
$\mathbb{Z} = \mathbb{N} \cup \{-n \colon n \in \mathbb{N}\}$

Il existe une bijection entre $\{-n \colon n \in \mathbb{N}\}$ et $\mathbb{N}$:

$$\begin{aligned} &f : \{-n \colon n \in \mathbb{N}\} \to \mathbb{N} \\ &f(x) = -x \end{aligned} \tag{4.5}$$

L'ensemble $\{-n : n \in \mathbb{N}\}$ est donc dénombrable. Donc, par application directe du Theorem 4.3 → p.9, $\mathbb{N} \cup \{-n : n \in \mathbb{N}\}$ est dénombrable, cela revient à dire que $\mathbb{Z}$ est dénombrable.

De même, $\mathbb{Q} = \mathbb{Q}^+ \cup \{-q : q \in \mathbb{Q}^+\}$. On suppose $\mathbb{Q}^+$ dénombrable (vu en cours). Il existe une bijection entre $\mathbb{Q}^+$ et $\mathbb{N}$:

$$f : \mathbb{Q}^+ \to \mathbb{N} \tag{4.6}$$
$$f(x) = -x \tag{4.7}$$

Donc $\{-q : q \in \mathbb{Q}^+\}$ est aussi dénombrable. En effet :

$$g : \{-q : q \in \mathbb{Q}^+\} \to \mathbb{N} \tag{4.8}$$
$$g(x) = f(-x) \tag{4.9}$$

est une bijection.
L'ensemble $\{-q : q \in \mathbb{N}\}$ est donc dénombrable. Donc, par application directe du Theorem 4.3 → p.9, $\mathbb{Q}^+ \cup \{-q : q \in \mathbb{Q}\}$ est dénombrable, cela revient à dire que $\mathbb{Q}$ est dénombrable.

Exercice 2 - Indénombrabilité de $\mathbb{R}$

1. En raisonnant par l'absurde, montrer que l'ensemble des nombres réels $]0.1[$ n'est pas dénombrable. Penser au schéma vu en cours et à la décomposition de tout réel sous forme décimale, $a = 0, a_1 a_2 a_3 \ldots$:

$$\begin{array}{c|cccccc} a_0 & 0, & a_{00} & a_{01} & a_{02} & \cdots & a_{0j} & \cdots \\ \vdots & & & \ddots & & \ddots & & \\ a_i & 0, & a_{i0} & a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots \\ \vdots & & & \ddots & & \ddots & & \\ a_j & 0, & a_{j0} & a_{j1} & a_{j2} & \cdots & a_{jj} & \cdots \\ \vdots & & & \ddots & & \ddots & & \end{array}$$

Considérons un réel $b = 0, b_1 b_2 \ldots b_j \ldots$ tel que pour tout $j \in \mathbb{N}$ par $b_j \neq a_{jj}$. Alors, par définition, $b \neq a_j$ pour tout $j \in \mathbb{N}$. C'est impossible, car les a_j sont censés énumérer tous les nombres réels. Donc l'ensemble $]0; 1[$ est indénombrable.

2. Montrer que les intervalles $]0; 1[$ et $]-\frac{\pi}{2}; \frac{\pi}{2}[$ ont même cardinal : $\|]0; 1[\| = \|]-\frac{\pi}{2}; \frac{\pi}{2}[\|$.

Considérons les fonctions :

$$\begin{aligned} f &: \left]-\frac{\pi}{2}; \frac{\pi}{2}\right[\to]0; \pi[\\ f(x) &= x + \frac{\pi}{2} \end{aligned} \tag{4.10}$$

$$\begin{aligned} g &:]0; \pi[\to]0; 1[\\ g(x) &= \frac{x}{\pi} \end{aligned} \tag{4.11}$$

Les fonctions f et g sont bijectives (pour le prouver, on peut regarder leurs fonctions inverses), donc $f \circ g$ est bijective entre $]0;1[$ et $]-\frac{\pi}{2};\frac{\pi}{2}[$. Ainsi $]0;1[$ et $]-\frac{\pi}{2};\frac{\pi}{2}[$ ont même cardinal.

3. En utilisant les fonctions bijectives bien connues de $\left]-\frac{\pi}{2};\frac{\pi}{2}\right[$ dans $\mathbb{R}$, montrer que $\mathbb{R}$ est indénombrable.

Il suffit de prendre la fonction tan. On a démontré dans la fonction précédente que l'on avait montré une bijection entre $]0;1[$ et $\mathbb{R}$ et ces ensembles ont même cardinalité. Comme on a montré à la question 1 que $]0;1[$ est indénombrable, $\mathbb{R}$ l'est aussi.

Exercice 3 - Cardinalité de $\mathbb{R}$

> **Note.** Le sujet que j'ai récupéré ne corresponds pas au sujet présent sur *Moodle* après la rédaction de ce TD. Certaines questions ont donc légèrement changé pour les exercices 3 et 4.

On admet que tout nombre réel $a \in \mathbb{R}$ est tel que $a = sup\{q \in \mathbb{Q} : q < a\}$ ($supE$ est la borne supérieure de l'ensemble non vide E.

1. Montrer qu'il existe une injection de $\mathbb{R}$ dans $\mathcal{P}(\mathbb{N})$ (ce qui s'écrit avec les notations vues en cours $\|\mathbb{R}\| \leq \|\mathcal{P}(\mathbb{N})\|$)

Considérons la fonction :

$$\begin{aligned} &f : \mathbb{R} \to \mathcal{P}(\mathbb{Q}) \\ &f(x) = \{q \in \mathbb{Q} : q < x\} \end{aligned} \tag{4.12}$$

Cette fonction est une injection si :

$$f(x) = f(x') \tag{4.13}$$

Alors :

$$\{q \in \mathbb{Q} : q < x\} = \{q \in \mathbb{Q} : q < x'\} \tag{4.14}$$

Donc :

$$sup\{q \in \mathbb{Q} : q < x\} = sup\{q \in \mathbb{Q} : q < x'\} \tag{4.15}$$

c'est-à-dire $x = x'$ d'après la propriété admise de $\mathbb{R}$. On a ainsi montré qu'il existe une injection f de $\mathbb{R}$ dans $\mathcal{P}(\mathbb{Q})$. On a montré dans l'**Exercice 1** que $\mathbb{Q}$ était dénombrable. Il existe une bijection g_0 de $\mathbb{Q}$ vers $\mathbb{N}$. Donc, il existe une bijection :

$$\begin{aligned} &g : \mathcal{P}(\mathbb{Q}) \to \mathcal{P}(\mathbb{N}) \\ &g(X) = \{g_0(x) : x \in X\} \end{aligned} \tag{4.16}$$

Ainsi, la fonction $f \circ g$ est une injection de $\mathbb{R}$ sur $\mathcal{P}(\mathbb{N})$.

2. Montrer qu'il existe une bijection de $\mathcal{P}(\mathbb{N})$ dans l'ensemble des suites numériques de 0 et 2.

Considérons la fonction :

$$\begin{aligned} &f\colon \mathcal{P}(\mathbb{N}) \to \{0,2\}^{\mathbb{N}} \\ &f(x) = (u_n)_{n\in\mathbb{N}} \end{aligned} \tag{4.17}$$

$$\text{avec } u_n = \begin{cases} 2, & \text{si } n \in x \\ 0, & \text{si } n \notin x \end{cases} \tag{4.18}$$

La fonction f est bien une bijection (sa fonction inverse est triviale).

3. L'ensemble de *Cantor* C est défini comme suit :

$$\mathsf{C} = \left\{ \sum_{n=0}^{\infty} \frac{u_n}{3^{n+1}} \colon u_n \in \{0,2\} \right\} \tag{4.19}$$

Il s'obtient en soustrayant a l'intervalle ferme $[0;1]$ les intervalles ouverts $]\frac{1}{3};\frac{2}{3}[$, $]\frac{1}{9};\frac{2}{2}[$, $]\frac{7}{9};\frac{8}{9}[$,*etc.*

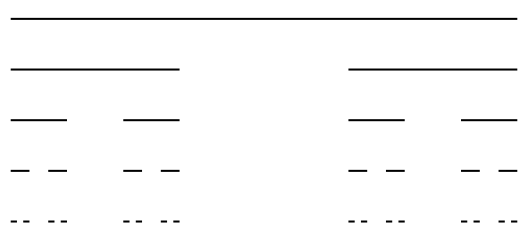

Figure 4.20 / Quatre premières itérations de la construction de l'ensemble de Cantor C

Montrer qu'il existe une bijection de l'ensemble des suites numériques de 0 et 2 dans$\mathsf{C}$.

$$\mathsf{C} := \left\{ \sum_{n=0}^{\infty} \frac{u_n}{3^{n+1}}, (u_n) \in \{0,2\}^{\mathbb{N}} \right\} \tag{4.21}$$

4. Déduire des deux questions précédentes qu'il existe une bijection entre C et $\mathcal{P}(\mathbb{N})$ (ce qui s'écrit avec les notations du cours $\|\mathsf{C}\| = \|\mathcal{P}(\mathbb{N})\|$) et qu'il existe une injection de $\mathcal{P}(\mathbb{N})$ dans $\mathbb{R}$

La fonction $g \circ f$ est une bijection entre $\mathcal{P}(\mathbb{N})$ et C. Par ailleurs, $\mathsf{C} \subseteq \mathbb{R}$. Par conséquent, $g \circ f$ est aussi une injection de $\mathcal{P}(\mathbb{N})$ dans $\mathbb{R}$.

5. En déduire que $\|\mathbb{R}\| = \|\mathcal{P}(\mathbb{N})\|$. Citer le théorème utilisé.

On a montré à la Question 1 qu'il existait une injection de $\mathbb{R}$ dans $\mathcal{P}(\mathbb{N})$ et montre aux Questions 2, 3 et 4 qu'il existait une injection de $\mathcal{P}(\mathbb{N})$ dans $\mathbb{R}$. Par

conséquent, d'après le Theorem 4.3 →p.9, il existe bien une bijection de $\mathbb{R}$ dans $\mathcal{P}(\mathbb{N})$, c'est-a-dire $\|\mathbb{R}\| = \|\mathcal{P}(\mathbb{N})\|$.

Cantor conjectura que tout ensemble de réels est soit au plus dénombrable, soit à la cardinalité de l'ensemble des réels (*i.e.* du continu). Exprime dans la théorie des ensembles de Zermelo et Fraenkel, cette conjecture s'appelle l'*Hypothèse du Continu* et s'écrit$\aleph_1 = 2^{\aleph_0}$ où $\aleph_1$ est le cardinal successeur de $\aleph_0$ puisque $\|\mathbb{R}\| = \|\mathcal{P}(\mathbb{N})\| = 2^{\aleph_0}$

Exercice 4 - Cardinalité de $\mathbb{C}$

1. Montrer qu'il existe une injection de $]0;1[^2$ dans $]0;1[$. on utilisera pour cela la représentation décimale des nombres réels (en choisissant la représentation la plus grande lexicographiquement d'un nombre réel quand elle existe, comme pour 0,5000 a la place de 0,999...). Il faudra définir un nombre réel c à partir de deux nombres réels $a = 0, a_1a_2a_3\ldots$ et $b = a = 0, a_1a_2a_3\ldots$.

Considérons la fonction :

$$\begin{aligned} &f\colon]0;1[^2 \to]0;1[\\ &\text{si :} \\ &\quad a = 0, a_1a_2a_3\ldots \text{ et } b = 0, b_1b_2b_3\ldots \\ &\text{alors :} \\ &\quad f(a,b) := \begin{cases} a_{\frac{n+1}{2}}, & \text{si } n \text{ est impair} \\ b_{\frac{n}{2}}, & \text{si } n \text{ est pair} \end{cases} = 0, a_1b_1a_2b_2\ldots \end{aligned} \tag{4.22}$$

La fonction f est une injection si :

$$0, a_1b_1a_2b_2\cdots = 0, a'_1b'_1a'_2b'_2\ldots \tag{4.23}$$

alors :

$$0, a_1a_2a_3\cdots = 0, a'_1a'_2a'_3\ldots \textbf{ et } 0, b_1b_2b_3\cdots = 0, b'_1b'_2b'_3\ldots \tag{4.24}$$

2. En déduire que $\|]0;1[^2\| = \|]0;1[\|$. Citer le théorème utilise.

On vient de définir f une injection de $]0;1[^2$ dans $]0;1[$. Réciproquement, on peut définir une injection :

$$\begin{aligned} &g\colon]0;1[\to]0;1[^2 \\ &g(x) = (x,x) \end{aligned} \tag{4.25}$$

Donc d'après le Theorem 4.3 →p.9, il existe une bijection entre $]0;1[$ et $]0;1[^2$.

3. En se servant des résultats de l'**Exercice 2**, montrer que $\|\mathbb{R}\|^2 = \|\mathbb{R}\|$. En déduire que les nombres complexes et les nombres réels ont la même cardinalité : $\|\mathbb{C}\| = \|\mathbb{R}\|$.

D'après l'**Exercice 2**, il existe une bijection f de $\mathbb{R}$ dans $]0;1[$. D'après la

question précédente, il existe une bijection g de $]0;1[^2$ dans $]0;1[$. Par conséquent, la fonction :

$$\begin{aligned} &h\colon \mathbb{R}^2 \to \mathbb{R} \\ &h = f^{-1} \circ g(f(x), f(y)) \end{aligned} \tag{4.26}$$

est une bijection. Donc $\|\mathbb{R}\|^2 = \|\mathbb{R}\|$.

La fonction de $\mathbb{C}$ dans $\mathbb{R}^2$ qui associe à tout nombre complexe la paire de nombre réels constitue de sa partie réelle et imaginaire est une bijection. Donc $\|\mathbb{C}\| = \|\mathbb{R}^2\|$. Donc si $\|\mathbb{R}\|^2 = \|\mathbb{R}\|$ donc Donc $\|\mathbb{C}\| = \|\mathbb{R}\|$.

4.2 Éxercices pratiques

References

Back-references to the pages where the publication was cited are given by •.

[1] Alonzo Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, **1936**.
DOI: 10.2307/2371045 (visited on 01/26/2025) 1, 4

[2] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **1937**.
DOI: 10.1112/plms/s2-42.1.230 (visited on 02/15/2025) 4

[3] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, **1953**.
DOI: 10.1090/s0002-9947-1953-0053041-6 (visited on 02/15/2025) 4