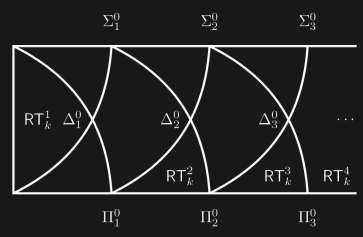
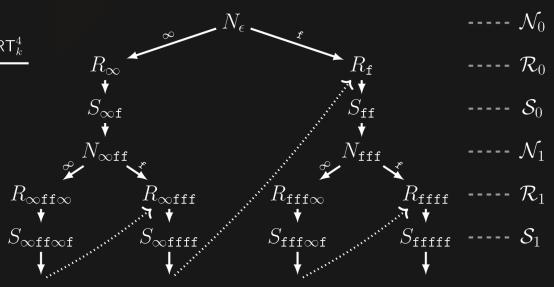
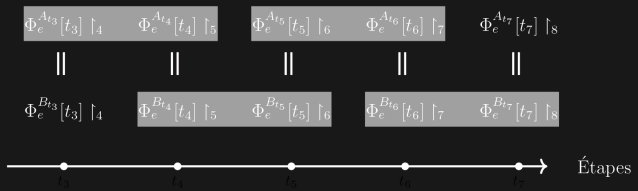
Tableau Noir – Calvage & Mounet







# calculabilité

DEGRÉS TURING

THÉORIE ALGORITHMIQUE DE L'ALÉATOIRE

MATHÉMATIQUES À REBOURS

HYPERCALCULABILITÉ


Benoît Monin

Ludovic Patey



C&M

TABLEAU NOIR

*Tableau Noir*

101. — Rached Mneimné. *Réduction des endomorphismes*
102. — Marc Hindry. *Arithmétique*
103. — Jean-Denis Eiden. *Géométrie analytique classique*
104. — Denis Choimet et Hervé Queffélec. *Analyse mathématique.– Grands théorèmes du vingtième siècle*
105. — Pascal Boyer. *Algèbre et géométries*
106. — Patrick Dehornoy. *La théorie des ensembles*
107. — Benoît Monin et Ludovic Patey. *Calculabilité*
108. — Gilles Godefroy. *Introduction aux méthodes de Baire* (à paraître)

Benoît Monin et Ludovic Patey

# Calculabilité

## Aléatoire, mathématiques à rebours et hypercalculabilité

Calvage & Mounet

Benoît Monin est maître de conférences à l'université de Créteil. Ses travaux portent sur la calculabilité classique, l'aléatoire algorithmique et l'hypercalculabilité.

benoit.monin@computability.fr
https://www.lacl.fr/%7Ebenoit.monin/

Ludovic Patey est chargé de recherche à l'université de Paris. Ses travaux portent sur la calculabilité classique et les mathématiques à rebours.

ludovic.patey@computability.fr
https://ludovicpatey.com/

Mathematics Subject Classification (2010) – Primary :

03-Dxx Computability and recursion theory
03D30 Other degrees and reducibilities in computability and recursion theory
03D60 Computability and recursion theory on ordinals, admissible sets, etc.
03D80 Applications of computability and recursion theory
03D10 Turing machines and related notions
03D20 Recursive functions and relations, subrecursive hierarchies
03D25 Recursively (computably) enumerable sets and degrees
03D32 Algorithmic randomness and dimension
03D55 Hierarchies of computability and definability
03D78 Computation over the reals, computable analysis
03B30 Foundations of classical theories (including reverse mathematics)
68-Qxx Theory of computing
68Q05 Models of computation (Turing machines, etc.)
68Q30 Algorithmic information theory (Kolmogorov complexity, etc.)
03B10 Classical first-order logic
03C07 Basic properties of first-order languages and structures
03F35 Second- and higher-order arithmetic and fragments
03F40 Gödel numberings and issues of incompleteness
03F60 Constructive and recursive analysis

ISBN 978-2-91-635296-1

*à nos familles*

# Table des matières

# Préambule

Ce livre est une introduction à la théorie de la calculabilité ainsi qu'à trois de ses ramifications principales, à savoir la théorie algorithmique de l'aléatoire, les mathématiques à rebours et l'hypercalculabilité. Il s'agit d'un ouvrage principalement destiné aux étudiants et enseignants en master de recherche en informatique et mathématiques, ainsi qu'aux chercheurs désirant acquérir une solide connaissance en calculabilité.

## Raison d'être du livre

Cet ouvrage trouve ses origines dans plusieurs constats.

- ▷ La littérature francophone sur la calculabilité classique est quasiment inexistante. Il existe quelques introductions à la calculabilité en français qui s'intègrent dans le cadre d'une présentation de la théorie de la complexité, comme l'excellent ouvrage d'Olivier Carton *Langages formels, calculabilité, et complexité* [28]. Les modèles de calcul sont alors présentés comme une base solide pour développer une théorie de la complexité algorithmique. Les résultats de calculabilité à proprement parler dépassent rarement l'indécidabilité du problème de l'arrêt et la définition de la hiérarchie arithmétique.
- ▷ Il existe de nombreux livres de référence en anglais sur la calculabilité (*Classical Recursion Theory : The Theory of Functions and Sets of Natural Numbers* de Piergiorgio Odifreddi [169], *Computability Theory* de Barry Cooper [41] ou encore *Turing computability : Theory and Applications* de Robert Soare [208]). Concernant la théorie algorithmique de l'aléatoire, on citera *Computability and Randomness* d'André Nies [167], et *Algorithmic Randomness and Complexity* de Rodney

Downey et Denis Hirschfeldt [50]. En mathématiques à rebours, la référence historique est *Subsystems of Second Order Arithmetic* de Stephen Simpson [203]. On mentionnera l'ouvrage plus récent de Denis Hirschfeldt, *Slicing the Truth* [87], et l'ouvrage en cours de rédaction, *Reverse Mathematics* de Damir Dzhafarov et Carl Mummert. Enfin, en hypercalculabilité, les deux références sont *Higher recursion theory* de Gerald Sacks [192] et *Recursion Theory : Computational Aspects of Definability* de Chi Tat Chong et Liang Yu [35]. Chacun de ces ouvrages présente l'état de l'art de la recherche pour un sous-domaine spécifique de la calculabilité, mais il n'existe pas un unique livre qui organise une présentation cohérente de ces différents aspects.

- Le calculabilité est un sujet d'étude extrêmement vaste, comme le laisse entrevoir la profusion d'articles sur le sujet et la taille des ouvrages de référence en anglais. Cependant, ce domaine reste très peu représenté en France, aussi bien du point de vue de la recherche que de l'enseignement. Peu de cours de calculabilité sont proposés en master, et leur contenu tend à véhiculer l'idée fausse que la calculabilité est une théorie des modèles de calcul. Cet ouvrage a pour ambition de donner ses lettres de noblesse à la calculabilité en France en donnant un petit panorama de sa grande richesse, encore trop méconnue.

## Organisation du livre

Ce livre est structuré en quatre grandes parties, à savoir la calculabilité classique, l'aléatoire algorithmique, les mathématiques à rebours et l'hypercalculabilité.

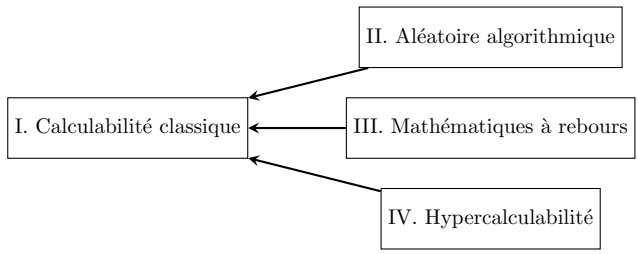
### Plan général

- La *calculabilité classique* est l'étude des degrés Turing, autrement dit la puissance calculatoire des ensembles d'entiers naturels. Elle constitue le cœur historique de la calculabilité, et le socle épistémique sur lequel s'appuient les trois parties suivantes.
- L'*aléatoire algorithmique* utilise la calculabilité classique pour donner un cadre effectif à la théorie de la mesure, qui permet d'étudier individuellement les suites de bits dites « aléatoires ». Les hiérarchies induites par la calculabilité classique permettent de définir différents niveaux d'aléa, à la lumière desquels on peut, par exemple, réexaminer la signification de tel ou tel théorème probabiliste stipulant qu'une propriété est vraie presque partout.
- Les *mathématiques à rebours* forment un programme sur les fondements des mathématiques, qui vise à identifier les axiomes nécessaires pour

prouver les théorèmes mathématiques de la vie de tous les jours [1]. Elles reposent sur une théorie de base, $\mathsf{RCA}_0$, dont les axiomes capturent les mathématiques « calculables » grâce à une correspondance entre la calculabilité et la définissabilité par des formules logiques.

- ▷ L'*hypercalculabilité* étend la notion de calculabilité à un cadre plus général rejoignant la théorie des ensembles. De la même manière que les opérations arithmétiques élémentaires s'étendent aux ordinaux, les machines de Turing peuvent étendre leur temps de calcul, des entiers aux ordinaux, et manipuler ainsi de plus grandes classes de réels. Il s'agit de l'approche par « modèle de calcul » de l'hypercalculabilité, qui correspond, comme pour la calculabilité classique, à certaines classes logiques.

Les trois dernières parties s'appuient toutes fortement sur la calculabilité classique, mais sont relativement indépendantes entre elles, et peuvent être pour l'essentiel, lues dans un ordre quelconque :




Dépendances entre les quatre grandes parties du livre

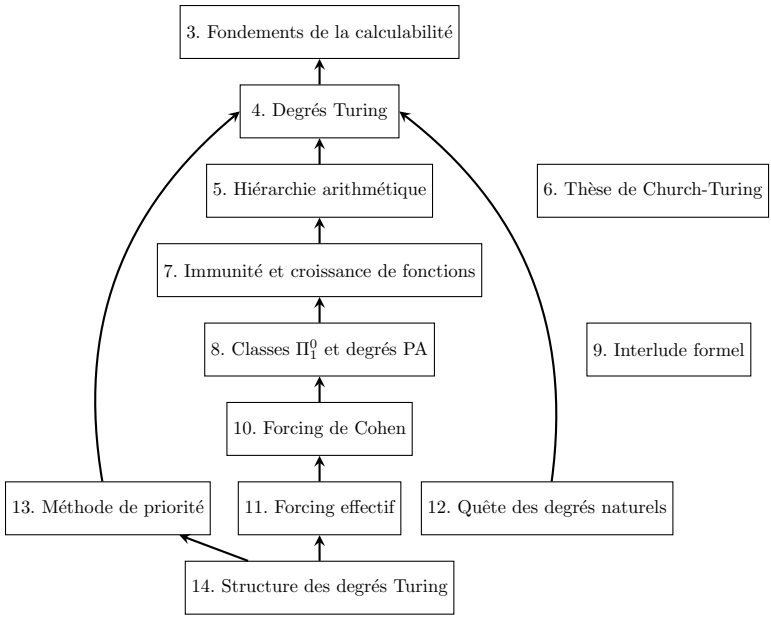
Notons tout de même la notion de « classe Borélienne » introduite dans la partie II, et fondamentale pour la compréhension de la partie IV.

### Calculabilité classique

La calculabilité classique a un rôle prépondérant en ce qu'elle fixe un cadre formel et une série d'outils qui serviront à développer les parties suivantes. Il convient donc de s'attarder sur la première partie et de détailler les dépendances de ses chapitres. Les chapitres fondamentaux sont principalement à lire linéairement, avec les exceptions suivantes : les chapitres 6 et 9 peuvent être lus indépendamment des autres, mais seront cependant utiles pour aborder sereinement la partie III du livre, sur les mathématiques à rebours. Les chapitres 12 et 14 ne seront pas absolument nécessaires pour la compréhension des parties suivantes, et visent à prendre un peu de recul

1. De la vie de tous les jours du mathématicien, s'entend.

sur notre travail. Le chapitre 12 — moins technique — le fera à travers l'examen d'une question spécifique, à la frontière de la philosophie ; et le chapitre 14 à travers une étude plus abstraite de la structure des degrés Turing, et la présentation de quelques-unes des grandes questions ouvertes du domaine.




Dépendances entre les chapitres de la calculabilité classique

# Comment lire ce livre

### Pour les enseignants

Ce livre peut servir de support pour un cours d'introduction à la calculabilité de niveau master, ainsi que pour des cours thématiques plus avancés, parlant de théorie algorithmique de l'aléatoire, de mathématiques à rebours et d'hypercalculabilité. Les sujets abordés vont bien au-delà des connaissances en calculabilité que l'on peut attendre d'un étudiant en master. Nous allons donc proposer un plan de cours contenant les notions incontournables.

L'équivalence entre les modèles de calcul forme un socle robuste pour le développement de la calculabilité. Cependant, les preuves peuvent sembler

assez longues et fastidieuses. De nos jours, avec la démocratisation des ordinateurs, on peut s'attendre à ce que les étudiants possèdent une certaine intuition de ce qu'est un algorithme, et il semble préférable de partir de cette intuition pour faire les premiers développements afin d'éviter un formalisme relativement lourd. Nous recommandons d'aborder l'équivalence des modèles de calcul, notamment entre les fonctions générales récursives et les machines de Turing, à travers une séance de travaux dirigés, où les étudiants auront l'occasion de manipuler les formalismes en définissant des fonctions de plus en plus compliquées pour se convaincre que ces définitions permettent de capturer tous les algorithmes.

Nous invitons nos lecteurs à suivre les développements des différents chapitres 3, 4, 5, 7, 8, 10, 13 dans cet ordre. Le chapitre 3 établit les premiers théorèmes fondamentaux de la calculabilité sur la base de l'intuition que l'on a des algorithmes. Le chapitre 4 définit la notion de machine à oracle, et de degré Turing. On y trouve les définitions les plus centrales de la calculabilité, comme le saut de Turing, et la méthode des extensions finies qui est une technique très puissante pour prouver l'existence de certains degrés Turing. Le chapitre 5 sur la hiérarchie arithmétique établit un lien essentiel entre la puissance calculatoire d'ensembles d'entiers et leur définissabilité par des formules de l'arithmétique, à travers le théorème de Post.

Avec le chapitre 7, on commence l'étude de différentes propriétés calculatoires fondamentales, comme la notion de degré hyperimmune, et ses liens avec l'existence de fonctions à croissance rapide. Cette étude est poursuivie dans le chapitre 8 sur les classes $\Pi^0_1$, où l'on définit la notion de degré PA qui est une notion centrale en calculabilité. On la retrouve tout au long de ce livre, notamment en aléatoire algorithmique et en mathématiques à rebours.

Le chapitre 10 introduit une technique fondamentale de la calculabilité, à savoir le forcing, présentée ici comme une élaboration de la méthode des extensions finies du chapitre 4. Ce chapitre peut également servir comme le premier niveau d'une compréhension graduelle de la technique générale du forcing de la théorie des ensembles.

Le chapitre 13 introduit finalement les méthodes de priorités, une autre technique fondamentale de la calculabilité, qui permet notamment de montrer l'existence de certains degrés calculatoirement énumérables.

### Pour les étudiants

Les compétences requises pour comprendre les notions présentées dans cet ouvrage sont celles d'une première année de licence en informatique ou mathématiques. Il est nécessaire pour l'essentiel de comprendre le langage

mathématique usuel (variables, quantifications, etc.), d'avoir quelques notions élémentaires de logique (preuve par contraposée, preuve par l'absurde, etc.), et de comprendre les éléments du corpus mathématique de base (comprendre ce qu'est une bijection, ce qu'est une intersection entre deux ensembles, les fonctions puissance et logarithme, etc.). En plus de cela, une expérience même sommaire en programmation, ou à défaut la compréhension de ce qu'est un algorithme, est également nécessaire pour aborder sereinement la lecture de ce livre.

Les mathématiques que nous utiliserons et qui ne sont pas enseignées dans une première année de licence seront introduites et expliquées au fur et à mesure des besoins (notions de base de topologie ou de théorie de la mesure par exemple). Ayant établi cela, notons tout de même que le degré d'élaboration des preuves, ainsi que la technicité de certains concepts, rendront sans doute cet ouvrage difficile à aborder sans une certaine maturité mathématique.

Les techniques développées en calculabilité sont assez différentes de celles apprises à travers un cursus mathématique standard. Cette particularité de la calculabilité est une force et rend cette discipline plus accessible puisqu'elle est peu sensible aux lacunes que l'on peut avoir développé au cours de son cursus (ou de son absence de cursus). En revanche, cette différence peut également déstabiliser l'étudiant car cela demande de se créer un univers conceptuel. Il va sans dire qu'en l'absence de professeur, il est d'autant plus essentiel de faire les exercices proposés au cours du livre pour bien intégrer les concepts. Les solutions sont disponibles à la fin du livre.

La taille de cet ouvrage peut s'avérer décourageante pour un étudiant désireux de faire ses premiers pas en calculabilité. Nous rappelons que ce livre couvre des connaissances allant bien au-delà de ce que l'on attend d'un étudiant en master. Nous recommandons donc aux autodidactes de suivre la suggestion de cours de la section précédente, destinée aux enseignants.

**Pour les chercheurs**

Cet ouvrage est une introduction à la calculabilité et à plusieurs de ses branches principales. Il ne faut cependant pas s'arrêter à l'aspect introductif de cet ouvrage, car la plupart des résultats présentés correspondent à l'état de l'art de la recherche. Ce livre s'adresse donc aussi bien aux chercheurs de domaines connexes désireux d'acquérir de solides connaissances en calculabilité, qu'aux doctorants et chercheurs se destinant à la recherche en calculabilité. En effet, les techniques et concepts de ce livre rendent directement accessibles les articles de recherche du domaine.

## Exercices

Les chapitres sont parsemés d'exercices de difficulté variable, dont la correction est donnée à la fin du livre. On ne rappellera jamais assez l'importance de faire des exercices pour bien assimiler les notions présentées dans les chapitres. L'intuition des concepts se crée en les manipulant sous toutes leurs formes. La difficulté des exercices est indiquée à l'aide d'un système d'étoiles ($\star$) allant de 0 à 3 : un exercice sans étoile est une application directe des définitions, tandis qu'un exercice à deux étoiles demande une profonde maîtrise des concepts pour être résolu. On trouvera également quelques exercices à trois étoiles, qui sont d'un niveau « recherche ».

À travers cet ouvrage, nous allons présenter beaucoup de propriétés calculatoires sur les ensembles d'entiers ou sur d'autres structures plus complexes. Outre les exercices donnés, il est important de faire preuve d'une curiosité intellectuelle consistant à chercher systématiquement comment se combinent ces propriétés, savoir si l'on peut construire des objets satisfaisant plusieurs d'entre elles simultanément, et ainsi de suite. De même, lorsque les théorèmes possèdent des hypothèses, il est utile de chercher des contre-exemples sans ces hypothèses, afin de mieux comprendre leur nécessité ainsi que leur usage dans la preuve.

## Errata

Il est impossible d'écrire un livre de cette taille sans laisser se glisser un certain nombre de coquilles. Cet ouvrage ne dérogera probablement pas à cette règle. Nous maintiendrons une liste des coquilles sur la page web des auteurs. Vous pouvez signaler les erreurs à l'une des adresses suivantes :

`benoit.monin@computability.fr`
ou
`ludovic.patey@computability.fr`

# Remerciements

Notre tout premier remerciement va à l'Agence nationale de la recherche, qui a en grande partie financé la collaboration entre les deux auteurs, notamment pendant l'écriture du livre, dans le cadre du projet « Aspects Calculatoires des Théorèmes Combinatoires – ACTC »

https://anr.fr/Projet-ANR-19-CE48-0012

Plusieurs résultats présentés dans ce livre, notamment le théorème 25-3.23 sur le principe des tiroirs ou la preuve simplifiée du théorème 25-3.24 de Liu proviennent d'articles financés par ce même projet.

Nous tenons également à remercier nos équipes et organismes de rattachement, qui nous ont fourni un vivier d'émulation intellectuelle ainsi qu'un soutien moral et financier. Pendant la rédaction de cet ouvrage, Monin était maître de conférences dans le Laboratoire d'Algorithmique, Complexité et Logique de l'Université Paris-Est Créteil et Patey chargé de recherche au CNRS, dans l'équipe Algèbre, Géométrie, Logique de l'Institut Camille Jordan à Villeurbanne.

De nombreux collègues nous ont généreusement aidé à améliorer la qualité de cet ouvrage, en donnant un regard critique sur son contenu scientifique et pédagogique, fort de leurs expertises respectives, ou bien en signalant les erreurs typographiques qui se sont inévitablement glissées dans le livre. Nous remercions donc Paul-Elliot Anglès d'Auriac, Sébastien Tavenas, Pascal Vanier, Mathieu Hoyrup, Benjamin Hellouin, Laurent Bienvenu, Denis Kuperberg, Julien Cervelle, Damir Dzhafarov, Loïc Gassmann, William Gaudelier, Denis Hirschfeldt, Quentin Le Houerou, Alexander Shen, Keita Yokoyama, Adrien Deloro, Pascal Monin et Shahin Amini.

Le contenu scientifique du livre est avant tout l'œuvre de la communauté de la calculabilité. Les auteurs ont forgé leurs intuitions en lisant les ouvrages

de leurs prédécesseurs et en ajoutant leur pierre à ce magnifique édifice intellectuel. Nous tenons à remercier nos collègues au niveau international pour les collaborations et les visites mutuelles ayant permis d'améliorer notre compréhension du sujet.

Nous tenons également à remercier Laurent Bienvenu, qui par son travail et à travers la direction de nos thèses respectives, a su nous transmettre sa passion, et a largement contribué à introduire la calculabilité en France.

La rédaction d'un ouvrage de cette ampleur prend beaucoup de temps et d'énergie, et n'aurait pu avoir lieu sans le soutien moral de nos familles et amis.

Nous tenons enfin à remercier les éditions Calvage et Mounet pour leur confiance, leur soutien et leur travail éditorial, et particulièrement Rached Mneimné pour sa relecture minutieuse du livre.

Chapitre 1

# Introduction

> Le savant n'étudie pas la nature parce que cela est utile ; il l'étudie parce qu'il y prend plaisir et il y prend plaisir parce qu'elle est belle. Si la nature n'était pas belle, elle ne vaudrait pas la peine d'être connue, la vie ne vaudrait pas la peine d'être vécue. Je ne parle pas ici, bien entendu, de cette beauté qui frappe les sens, de la beauté des qualités et des apparences ; non que j'en fasse fi, loin de là, mais elle n'a rien à faire avec la science ; je veux parler de cette beauté plus intime qui vient de l'ordre harmonieux des parties, et qu'une intelligence pure peut saisir.
>
> Science et méthode, Henri Poincaré

**Qu'est-ce que la calculabilité ?** On considère classiquement la calculabilité comme l'un des quatre piliers de la logique, aux côtés de la théorie des ensembles, la théorie des modèles et la théorie de la preuve. Le domaine s'est au départ forgé sur la question de ce qui caractérise les fonctions $f : \mathbb{N} \to \mathbb{N}$ dont les valeurs peuvent être obtenues par un processus purement *mécanisable* ou *algorithmique*, en un temps fini, bien qu'arbitrairement grand. Nous dirons que de telles fonctions sont *effectivement calculables*. Bien avant l'apparition des premiers ordinateurs, la calculabilité a fondé sa base théorique sur un constat — ou plutôt un miracle — à savoir l'existence d'une définition robuste, consensuelle et indépendante de tout formalisme, de la notion épistémologique de fonction effectivement calculable.

La question initiale — à savoir « Qu'est-ce qu'une fonction calculable ? » — ayant obtenu une réponse satisfaisante, l'étude s'est naturellement portée vers la question de savoir, parmi les fonctions naturelles, lesquelles sont calculables et lesquelles ne le sont pas. Par la suite, le domaine a connu un développement considérable grâce à la notion de calculabilité relative, la question n'étant plus de déterminer si une fonction est calculable ou non, mais d'identifier la puissance calculatoire intrinsèque à cette fonction, à travers des questions comme « Si cette fonction était calculable, quelles autres fonctions pourrait-on calculer ? »

Plus récemment, le sujet d'étude s'est étendu à de très nombreux objets mathématiques — par exemple des structures algébriques ou des sous-ensembles de $\mathbb{R}$ — et a donné de nombreuses ramifications. Nous verrons notamment dans ce livre que la calculabilité sert de fondement robuste à la théorie algorithmique de l'aléatoire, et aux mathématiques à rebours, dont les objets d'études sont les théorèmes mathématiques eux-mêmes.

De nos jours, l'appellation « calculabilité » pour un domaine qui étudie des objets mathématiques arbitraires, dont la plupart ne sont pas calculables, peut sembler étonnante, voire un reliquat de son sujet d'étude historique. En vérité, ce nom est toujours judicieux, mais sa signification a changé : le terme *calculabilité* ne porte plus sur le sujet de l'étude, mais sur l'angle sous lequel le sujet est abordé. Une définition moderne de la calculabilité en une phrase pourrait être : **la calculabilité est l'étude des mathématiques sous le prisme de leur complexité calculatoire**.

## 1. Qu'est-ce qu'une fonction calculable ?

La principale difficulté de cette question réside dans l'obtention d'une classe de fonctions suffisamment robuste pour ne pas dépendre du modèle d'ordinateur, du choix du langage de programmation, des progrès technologiques, ou de l'avancée de la connaissance de manière générale.

Avec l'avènement des ordinateurs, la notion d'algorithme s'est peu à peu ancrée dans la culture scientifique. Toute personne ayant déjà eu un premier contact avec la programmation se sera déjà formée une bonne idée de ce qu'est une tâche automatisable. Forts de notre connaissance de l'informatique, la définition suivante viendrait naturellement : « Une fonction est effectivement calculable si elle possède un algorithme, autrement dit si elle peut être programmée dans un langage suffisamment expressif, et exécutée par un ordinateur suffisamment puissant. »

Cette définition, si elle a l'avantage d'être en adéquation avec notre intuition, ne fournit pas un cadre suffisamment formel pour raisonner sur la classe des fonctions calculables. Une seconde approche consisterait à

fixer un ordinateur et un langage de programmation étalon, et définir une fonction comme calculable si elle est programmable dans ce langage, et exécutable par cet ordinateur en temps fini, à l'aide de suffisamment de mémoire. Si l'on ne se préoccupe pas de la rapidité d'exécution, ni de l'espace mémoire nécessaire, il apparaît rapidement que cette définition coïncide avec la précédente. En effet, la puissance et la mémoire des ordinateurs augmentent au gré des progrès technologiques, et permettent donc d'exécuter plus rapidement les programmes, mais n'augmentent pas pour autant la classe des fonctions calculables. Même les ordinateurs basés sur de nouveaux paradigmes de calcul, comme les ordinateurs quantiques ou biologiques, sont simulables — au prix d'un surcoût d'espace et de temps exponentiel — par des ordinateurs classiques, et ne changent donc pas la classe des fonctions calculables. Quant aux langages de programmation, l'existence de systèmes d'exploitation et d'interpréteurs permettent de se convaincre aisément que les principaux d'entre eux tels que C++, Java ou Python, permettent de programmer — plus ou moins élégamment — les mêmes fonctions mathématiques. Cela montre donc empiriquement une certaine robustesse dans la définition de la classe des fonctions programmables.

Un problème subsiste : quelle est la garantie que les ordinateurs actuels représentent la limite de ce qui est automatisable, ou calculable par un être humain ? Qui nous dit qu'avec les progrès de la science, nous ne découvrirons pas un nouveau paradigme de calcul ou une nouvelle manière de raisonner permettant de considérer comme calculable une plus large classe de fonctions ? C'est là le sujet d'une longue quête fondationnelle débutée au XX$^{\text{e}}$ siècle, et aboutissant à la fameuse thèse de Church-Turing en 1936, que nous présenterons dans le chapitre 6.

## 2. Quelles sont les fonctions incalculables ?

Au regard de notre définition précédente, pour l'instant très informelle, la plupart — si ce n'est la totalité — des fonctions mathématiques utilisées au quotidien sont calculables : l'addition, la multiplication, la fonction $(n, m) \mapsto n^m$, la fonction qui à $n$ associe le $n$-ième nombre premier, ou encore celle qui calcule le plus grand diviseur commun de deux entiers naturels, sont toutes calculables. On peut rajouter à cette liste des exemples moins triviaux : la fonction qui prend un programme informatique écrit en C++ et détermine si le programme est syntaxiquement correct — c'est ce que fait entre autres choses un compilateur pour C++ — ou encore celle qui renvoie la $n$-ième décimale de $\pi$, $\sqrt{2}$ ou du nombre d'or — chacun de ces nombres est la somme d'une suite calculable de rationnels de convergence

suffisamment rapide — ou pour finir celle qui à $n$ associe le nombre de parties possibles que l'on peut faire au jeu de go sur un plateau — appelé aussi *goban* — de taille $n \times n$. Ce dernier exemple illustre en particulier le fait suivant : on ne s'occupe pas en calculabilité du temps que prend un calcul. Seule l'existence d'un algorithme nous importe. Dans le cas du nombre de parties au jeu de go, l'algorithme en question repose sur une idée simple ; il « suffit » de lister toutes les parties possibles et de les compter. Cependant, le temps d'exécution d'un tel algorithme est tellement grand que cela le rend impossible à utiliser en pratique pour $n > 2$ ([1]). Pour $n = 19$, qui est la taille d'un goban standard, ce nombre est compris entre $10^{10^{48}}$ et $10^{10^{171}}$ [225], ce qui fait clairement trop de parties à compter même si tous les ordinateurs de monde s'y attelaient pour un milliard d'années…

Même si la fonction de multiplication par 2 nous est, en un sens, bien plus accessible que celle qui compte le nombre de parties au jeu de go, il existe un algorithme qui calcule chacune d'entre elles. Ces deux fonctions ne sont donc pas différentes l'une de l'autre du point de vue de la calculabilité : elles sont toutes les deux calculables, et nous allons principalement nous intéresser aux fonctions qui *ne le sont pas*, c'est-à-dire les fonctions dont les valeurs *ne peuvent pas* être obtenues par un processus purement mécanisable ou algorithmique. La simple existence de telles fonctions n'est pas une évidence en soi, et l'une des premières tâches à laquelle nous nous attellerons sera d'en montrer l'existence. Cela sera fait dans le chapitre suivant via l'argument diagonal de Cantor. Nous donnerons ensuite tout au long du livre de très nombreux exemples de telles fonctions, la plus connue d'entre elles étant sans doute le *problème de l'arrêt*, défini comme la fonction qui prend en entrée un programme, et détermine si son exécution va s'arrêter, en temps fini nécessairement. Nous verrons que le problème de l'arrêt n'est pas calculable ; et il est important de comprendre qu'il s'agit bien ici d'une impossibilité théorique et fondamentale, qui ne dépend pas de la puissance ou vitesse de calcul des ordinateurs. L'incalculabilité du problème de l'arrêt n'est pas due à une ignorance de son algorithme qui pourrait être un jour découvert, mais bien à une impossibilité absolue, car l'existence d'un tel algorithme entraînerait un paradoxe.

## 3. Motivations

La calculabilité porte principalement sur l'étude des fonctions — ou objets mathématiques plus généraux — incalculables. Il est légitime de se demander si une telle étude est bien raisonnable. Si même certaines fonctions calculables nous sont inaccessibles — comme le nombre de parties possibles

1. Il y a déjà 386 356 909 593 parties possibles sur un goban de taille $2 \times 2$ [225] !

au jeu de go — alors à quoi bon se donner la peine de réfléchir sur des fonctions *encore plus inaccessibles* ?

Une première motivation pour notre étude est d'ordre exploratoire. Il existe des objets inaccessibles, essayons d'en explorer l'univers. Simplement parce qu'il est là, et par curiosité sur les mystères qu'il renferme. Nos efforts seront récompensés par une série de théorèmes d'une très grande profondeur. Quiconque se plonge avec sérieux dans les développements de ce livre, une fois peut-être passé quelques difficultés d'adaptation inhérentes à toute discipline scientifique, verra un monde d'une richesse stupéfiante prendre vie dans son esprit, avec sa faune et sa flore, ses règles et mécanismes. La calculabilité est caractérisée par la nature très dynamique de ses preuves, chacune d'elles en offrant un aperçu sur le fonctionnement détaillé d'un fragment de la machinerie titanesque qui anime cet univers.

Une deuxième motivation survient tout simplement par nécessité. Les Pythagoriciens se sont retrouvés contraints et forcés d'admettre l'existence de mesures irrationnelles, comme la diagonale d'un carré de côté 1, ce qui allait à l'encontre de leur compréhension du monde, qu'ils pensaient explicable uniquement en se fondant sur les rapports entre nombres entiers. Mais si l'on admet l'existence des entiers, et l'existence du carré, on est forcé d'admettre celle de quantités *incommensurables*, que l'on appelle aujourd'hui irrationnelles, comme $\sqrt{2}$. De la même manière, nous verrons que si l'on admet l'existence des objets calculables, on est forcé aussi d'admettre l'existence d'objets qui ne le sont pas, et qui apparaissent malgré tout naturellement dans toute une série de situations.

Une dernière motivation enfin est d'ordre pratique. La calculabilité, via la compréhension qu'elle donne des objets incalculables, a obtenu des succès majeurs en fournissant un cadre formel pour l'étude de questions à la frontière entre science et philosophie. Nous en verrons deux : la recherche de la définition d'objets aléatoires avec la partie II, et la compréhension de ce que signifie *la force* d'un théorème, notamment par rapport à un autre, avec la partie III. La partie IV de ce livre amènera quant à elle la calculabilité vers la frontière qu'elle partage avec la théorie des ensembles.

# 4. Panorama de la calculabilité

La calculabilité peut se décomposer en plusieurs sous-domaines, qui s'appuient tous sur la même notion robuste de fonction effectivement calculable.

## 4.1. Domaines couverts par ce livre

Cet ouvrage est décomposé en quatre parties, chacune d'entre elles couvrant une ramification de la calculabilité : la calculabilité classique, l'aléatoire algorithmique, les mathématiques à rebours, et l'hypercalculabilité.

### Calculabilité classique

Comme nous l'avons spécifié, la calculabilité porte avant tout sur des objets que l'on ne peut pas calculer. La calculabilité classique se concentre sur les fonctions $f : \mathbb{N} \to \mathbb{N}$ ainsi que sur les ensembles d'entiers $E \subseteq \mathbb{N}$. Remarquons qu'un tel ensemble peut aussi être représenté par sa fonction caractéristique $\chi_E : \mathbb{N} \to \mathbb{N}$ définie par $\chi_E(x) = 1$ si $x \in E$, et $\chi_E(x) = 0$ sinon.

Les développements de la calculabilité classique s'articulent autour d'un outil fondamental qui nous permettra de comparer ou encore de mesurer le *degré d'incalculabilité* d'une fonction, appelé aussi *degré d'insolubilité* ou encore *degré Turing*, en référence au mathématicien Alan Turing qui introduisit la notion. Fixons une fonction non calculable $g : \mathbb{N} \to \mathbb{N}$. Il est naturel de se demander « Si j'étais capable de calculer $g$, quelles autres fonctions pourrais-je calculer ? » On dit qu'une fonction $f : \mathbb{N} \to \mathbb{N}$ est *calculable relativement à* $g$ (ou $g$-calculable) s'il existe un algorithme permettant de calculer $f$ dans un langage de programmation étendu, où l'on aurait rajouté la fonction $g$ comme primitive : une instruction spéciale nous permet d'appeler la fonction $g$ sur un paramètre $n$ dans notre programme, comme si elle existait réellement, et d'en récupérer le résultat. Si $f$ est $g$-calculable, rien ne nous indique comment calculer $g$, mais si l'on disposait d'un « oracle » nous permettant de calculer les valeurs de $g$, il serait possible de calculer les valeurs de $f$.

Cette notion de calculabilité relative nous permet de définir un pré-ordre partiel entre les fonctions, notant $f \leqslant_T g$ si la fonction $f$ est $g$-calculable. Il s'agit de la *réduction Turing*. Différentes fonctions peuvent porter la même puissance calculatoire, au sens où elles sont mutuellement calculables. On définit donc le *degré Turing* d'une fonction $f : \mathbb{N} \to \mathbb{N}$ l'ensemble $\deg_T f$ de toutes les fonctions $g$ telles que $f \leqslant_T g$ et $g \leqslant_T f$. La notion de degré Turing représente une puissance calculatoire, au sens où deux fonctions de même degré Turing sont indistinguables du point de vue de la calculabilité. Le pré-ordre partiel sur les fonctions induit un ordre partiel sur les degrés Turing.

La calculabilité classique porte principalement sur l'étude des degrés Turing munis de la relation d'ordre partielle définie ci-dessus. Existe-t-il une infinité de puissances calculatoires ? Sont-elles linéairement ordonnées ? Plus généralement, quelles sont les propriétés de cet ordre partiel ? Il s'avère que cette structure est extrêmement riche et complexe, comme nous aurons l'occasion de le voir.

### Aléatoire algorithmique

La théorie classique des probabilités étudie les phénomènes probabilistes, modélisés avec succès via la notion de mesure qui sert à définir formellement

les lois de probabilité. Cette théorie n'a en revanche pas les outils nécessaires — et ce n'est pas là son objectif — pour parler d'objets aléatoires *individuellement*. C'est ce point précis que la théorie algorithmique de l'aléatoire se propose d'éclaircir, en s'appuyant sur la calculabilité. Voyons à travers un exemple de quoi il s'agit.

Représentons un nombre réel $R \in [0, 1]$ par son développement binaire, de la forme $R = 0.b_0b_1b_2b_3\ldots$ où $(b_n)_{n\in\mathbb{N}}$ est une suite de bits. Supposons que le réel $R$ soit obtenu en tirant ses bits *au hasard* par une suite de tirage à pile ou face. On suppose bien entendu que chaque tirage est *équiprobable* : nous avons une chance sur deux d'obtenir pile et une chance sur deux d'obtenir face. L'intuition nous dicte que le réel $R$ ainsi obtenu est *aléatoire*. Que cela signifie-t-il exactement ? On ne s'attend par exemple pas à n'obtenir que des « pile » sur les cent mille premiers lancers : si chaque tirage est équiprobable, cela ne peut arriver, ou en tout cas avec une probabilité tellement faible que l'on peut la considérer comme négligeable. On ne s'attend pas non plus à obtenir deux fois plus de « pile » que de « face ». Là encore, la probabilité que cela arrive sur cent mille tirages est tellement faible que l'on supposera les tirages biaisés plutôt que d'être témoin d'un événement si improbable. On peut de fait identifier une première propriété que l'on est en droit d'attendre d'une suite de tirages équiprobables : la suite obtenue devrait respecter la loi des grands nombres, c'est-à-dire que les nombres de tirages « pile » et « face » doivent à peu près être les mêmes.

Cela est-il pour autant suffisant ? Supposons à présent par exemple que sur chaque tirage numéro $n$, si $n$ est un nombre premier on obtient systématiquement un « pile ». Dans l'hypothèse où une certaine obsession des nombres premiers nous conduirait à remarquer ce curieux phénomène, nous serons là encore en face d'une énigme — un peu absurde — et l'on sera amené à penser que d'une manière ou d'une autre, quelque chose d'anormal est en train de se produire. Mais prenons encore plus de recul. Au fond, et peu importe la suite de bit obtenue, on peut identifier des nombres $n_1 < n_2 < n_3 < \ldots$ tels que les tirages numéros $n_1$, $n_2$, $n_3$, $\ldots$ sont tous des tirages « pile ». Dans le cas où notre suite $n_1$, $n_2$, $n_3$, $\ldots$ contient les nombres premiers, cela nous semble relever d'un « bug probabiliste », mais pourquoi cela devrait-il être acceptable si $n_1$, $n_2$, $n_3$, $\ldots$ sont des nombres entiers quelconques ? C'est là que la calculabilité entre en jeu, et va nous permettre de formaliser précisément les propriétés que devrait avoir — en accord avec notre intuition humaine — une suite de bits aléatoire.

### Mathématiques à rebours

La notion de *théorème* est relative à un système d'axiomes. Lorsque l'on omet de mentionner le système de référence, il est communément admis que l'on se réfère au système de Zermelo Fraenkel (ZF), qui représente

un ensemble d'axiomes consensuels servant de fondement à l'ensemble des mathématiques. Le système ZF est cependant très puissant, et nous n'avons aucune garantie de son absence de contradiction.

Les mathématiques à rebours visent à trouver les axiomes nécessaires et suffisants pour prouver les théorèmes des mathématiques de tous les jours. Il s'agit donc d'étudier des théorèmes existants, pour en trouver des preuves plus élémentaires, ou au contraire pour montrer l'optimalité de leur preuve. Mieux comprendre les hypothèses des théorèmes permet de mieux maîtriser leur « fragilité » face à une potentielle contradiction du système de preuves. Il s'agit donc d'une démarche méta-mathématique visant à répondre à la question « Quelle confiance peut-on avoir en nos mathématiques ? »

Au premier abord, cette démarche n'est pas liée à la calculabilité. Cependant, les mathématiques à rebours se réfèrent à une théorie de base, $\mathsf{RCA}_0$, capturant les *mathématiques calculables*, et qui représente une base de confiance plus en lien avec le monde concret, car ses objets étant calculables, ils peuvent être représentés par un algorithme, donc possèdent une description finitaire. Les mathématiques à rebours consistent donc, étant donné un théorème $T$, à chercher des axiomes $A$ tels que $\mathsf{RCA}_0$ prouve l'équivalence entre $A$ et $T$. Par le choix de la théorie de base $\mathsf{RCA}_0$, les équivalences sont des procédés calculatoires faisant appel aux outils de la calculabilité.

**Hypercalculabilité**

Une des raisons du succès de la calculabilité en tant qu'outil d'analyse des mathématiques réside dans l'existence d'une solide intuition de la notion de calcul, permettant ainsi de guider la manipulation des concepts et de prouver des théorèmes sans s'embarrasser d'un lourd formalisme. L'hypercalculabilité vise à étendre la portée de ces outils à des modèles de calcul plus puissants, qui peuvent être vus comme des machines ayant la possibilité de poursuivre leur exécution pendant un temps de calcul infini (formellement en temps de calcul ordinal). Tout comme les notions de calculabilité classique peuvent être capturées par des formules logiques, il en va de même pour l'hypercalculabilité. Par exemple, là où les ensembles d'entiers que l'on peut énumérer (dans le désordre) par un programme informatique sont ceux qui peuvent être décrits par une formule dite $\Sigma^0_1$ de l'arithmétique, ceux qui sont énumérables par un programme informatique hypercalculable sont ceux qui peuvent être définis par une formule dite $\Pi^1_1$ de l'arithmétique.

Nous verrons que cet aspect des choses rapproche l'hypercalculabilité de la théorie descriptive des ensembles, une branche de la théorie des ensembles qui classifie ces derniers selon le degré de difficulté à les décrire. L'hypercalculabilité peut être vue comme un pont entre la théorie descriptive des ensembles et la calculabilité classique.

## 4.2. Autres branches de la calculabilité

Afin de permettre à cet ouvrage de conserver une taille raisonnable, nous avons fait le choix de faire l'impasse sur deux branches importantes de la calculabilité, à savoir la théorie des structures calculables et l'étude des degrés d'énumération.

### Théorie des structures calculables

Il s'agit d'une branche de la calculabilité qui étudie dans quelle mesure les propriétés algébriques d'une structure mathématique affectent leur complexité descriptive. Par structure, on entend des ensembles munis d'opérations, comme les groupes, les anneaux et les corps, mais également toute structure au sens de la théorie des modèles. Cette branche emprunte ses techniques à la fois à la théorie des modèles et à la calculabilité classique pour répondre à cette question.

Concrètement, cette théorie étudie des structures dénombrables et pose des questions de la forme « Étant donné une structure calculable $\mathcal{A}$, quels sont les degrés Turing possibles des structures isomorphes à $\mathcal{A}$ ? » ou bien « Étant donné deux structures calculables et isomorphes, de quelle puissance calculatoire a-t-on besoin pour calculer leur isomorphisme ? » Par exemple, les instances calculables des ordres denses sans extrémités sont toutes deux à deux calculatoirement isomorphes. On les appelle *calculatoirement catégoriques*.

### Degrés d'énumération

La calculabilité classique place « le calculable » comme puissance calculatoire de référence. Mais certains problèmes s'expriment de manière naturelle sous forme d'ensembles non calculables, dont les éléments peuvent toutefois être énumérés dans le désordre par un processus calculable. On appelle ces ensembles *calculatoirement énumérables* (c. e.). En particulier, si $E$ est un ensemble d'entiers c. e. et si $n \in E$, il est possible de s'en rendre compte en un temps fini, en lançant la procédure d'énumération et en attendant que $n$ apparaisse. En revanche, si $n \notin E$, alors il ne sera pas possible en général de le savoir en un temps fini. Par exemple, l'ensemble des équations diophantiennes (des équations à coefficients entiers, comme $3x^3 - 2y^2 + x - 2 = 0$) qui admettent des solutions entières est calculatoirement énumérable, car il suffit de chercher exhaustivement des solutions, et d'énumérer l'équation si une telle solution existe.

Les degrés d'énumération placent « le calculatoirement énumérable » comme puissance de référence. On peut définir une *réduction d'énumération* $A \leqslant_e B$ ssi toute énumération des éléments de $B$ calcule une énumération des éléments de $A$. Cette réduction est une pré-ordre partiel, qui induit une

notion de *degré d'énumération* : le degré d'énumération de $A$ est l'ensemble $\deg_e(A)$ de tous les ensembles $B$ tels que $A \leqslant_e B$ et $B \leqslant_e A$. L'étude des degrés d'énumération munis de l'ordre partiel $\leqslant_e$ constitue une branche active de recherche en calculabilité.

Chapitre 2

# Infinis de Cantor

Si l'on devait donner une date à la naissance de la logique moderne, nous situerions sans hésiter celle-ci en 1872, date à laquelle Georg Cantor expose sa première démonstration du théorème 4.1 à venir, où il établit la non-dénombrabilité des nombres réels. Cantor isolera plus tard la quintessence de cette première preuve à travers son fameux *argument diagonal*, qui aura une place centrale en calculabilité.

Georg Cantor, 1845–1918

Les travaux de Cantor sur l'infini marquent le début d'une théorie des ensembles « complexe », qui jouera un grand rôle dans la quête fondationnelle des mathématiques du début du XX$^{e}$ siècle, dont nous parlerons en détail en première partie du chapitre 9, sur la fameuse « crise des fondements ». Cette crise débouchera sur le développement de la logique mathématique telle que nous la connaissons aujourd'hui, avec la théorie des ensembles moderne, dite ZFC, mais aussi avec le développement des premières théories du calcul, utilisées par Gödel pour montrer son fameux théorème d'incomplétude, que l'on peut considérer comme une déclinaison sophistiquée de l'argument diagonal de Cantor.

De quoi s'agit-il exactement ? Cantor montre que les ensembles infinis n'ont pas tous la même « taille ». Il y a strictement plus de nombres réels que de nombres entiers, dans un sens que nous définirons précisément dans quelques lignes. Cantor se basera sur cette découverte pour développer une étude mathématique de l'infini. Il créera notamment les nombres transfinis, qui constitueront la colonne vertébrale des définitions mathématiques, et que nous aborderons dans le chapitre 27.

Cantor ne fut cependant pas le premier à remarquer que l'infini n'obéissait pas aux mêmes règles que le fini. En particulier, une caractéristique surprenante des ensembles infinis est que le tout n'est pas forcément plus grand que ses parties. Galilée en fait une exposition lumineuse dans son ouvrage « Discours concernant deux sciences nouvelles » [71] à travers un dialogue savoureux entre deux personnages, Salviati et Simplicio :

– Salviati. *J'estime que les épithètes comme « plus grand », « plus petit » et « égal » ne conviennent pas aux grandeurs infinies, dont il est impossible de dire que l'une est plus grande, plus petite ou égale à une autre. Mais voici pour le prouver un raisonnement qui me revient à l'esprit : vous savez parfaitement je suppose quels nombres sont carrés et quels nombres ne le sont pas.*
– Simplicio. *Je sais parfaitement qu'un nombre carré provient de la multiplication d'un autre nombre par lui-même ; ainsi quatre, neuf, etc. sont des nombres carrés résultant de la multiplication de deux, trois, etc. par eux-mêmes.*
– Salviati. *Fort bien, quant aux nombres qui ne proviennent pas de nombres multipliés par eux-mêmes, ce ne sont pas des carrés. Par conséquent, si je dis que les nombres pris dans leur totalité, en incluant les carrés et les non-carrés, sont plus nombreux que les carrés seuls, j'énoncerai, n'est-ce pas, une proposition vraie ?*
– Simplicio. *Très certainement.*
– Salviati. *Si je demande maintenant combien il y a de nombres carrés, on peut répondre sans se tromper qu'il y en a autant que de racines correspondantes, attendu que tout carré a sa racine et toute racine son carré, qu'un carré n'a pas plus d'une racine et une racine pas plus d'un carré.*
– Simplicio. *Exactement.*
– Salviati. *Mais si je demande combien il y a de racines, on ne peut nier qu'il y en a autant que de nombres, puisque tout nombre est la racine de quelque carré il faudrait admettre que les carrés sont aussi nombreux que tous les nombres pris ensemble.*

On observe à la lecture du dialogue de Galilée le piège du paradoxe se refermer sur Simplicio. Galilée utilise pour cela un concept qui sera repris par Cantor : deux ensembles $A$ et $B$ « ont le même nombre d'éléments » si l'on peut faire correspondre exactement les éléments de $A$ et les éléments de $B$, autrement dit s'il existe une bijection entre les deux ensembles.

# 1. Équipotence et subpotence

Rappelons qu'une fonction $f : E \to F$ est *injective* si

$$\forall x, y \quad x \neq y \Rightarrow f(x) \neq f(y),$$

*surjective* si son image est l'ensemble $F$ tout entier, et *bijective* si elle est à la fois injective et surjective.

**Définition 1.1.** Deux ensembles $E$ et $F$ sont *équipotents* s'il existe une bijection entre eux. On écrira alors $|E| = |F|$. ♢

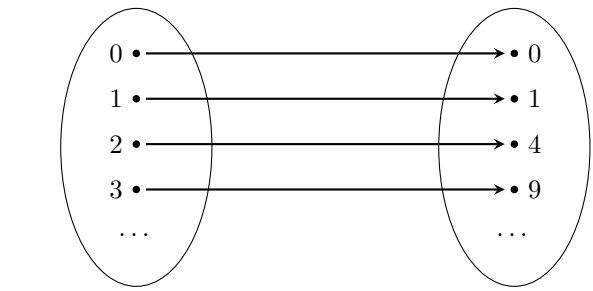



FIGURE 1.2 – *L'argument de Galilée pour dire qu'il y a « autant » d'entiers que de carrés*

Selon notre définition les entiers, et les carrés d'entiers ont donc la même cardinalité : il y a autant d'éléments dans les deux ensembles. Ce qui semble paradoxal, c'est que les carrés d'entiers forment une partie stricte de l'ensemble des entiers. Le paradoxe est résolu de la manière la plus simple qui soit : l'intuition que l'on a sur les ensembles finis, qui veut qu'une sous-partie stricte d'un ensemble contienne moins d'éléments n'est tout simplement plus vraie pour les ensembles infinis.

**Remarque**

La notation $|E| = |F|$ semble suggérer l'égalité entre deux objets $|E|$ et $|F|$, que l'on nomme respectivement *cardinalité* de $E$ et *cardinalité* de $F$. Il est possible de donner une définition précise de $|E|$, en tant qu'objet mathématique. Nous nous contenterons pour le moment de définir la cardinalité comme la notion informelle de taille d'un ensemble, et si l'objet $|E|$ n'est pas clairement défini, l'énoncé $|E| = |F|$ l'est, et suffit à notre traitement de l'infini.

Notons que Galilée utilise son idée pour expliquer justement pourquoi il n'y a aucun sens à comparer la taille des ensembles infinis. C'est là qu'intervient tout le génie de Cantor, qui découvrira que contrairement à l'intuition de Galilée, il est possible de donner une notion formelle de taille aux ensembles infinis : il existe des infinis « plus gros » que d'autres.

Intuitivement, un ensemble $F$ est au moins aussi gros qu'un ensemble $E$ si l'on peut faire correspondre aux éléments de $E$ des éléments distincts de $F$, autrement dit si l'on peut associer à chaque élément de $E$ un « représentant » qui lui est propre dans $F$.

**Définition 1.3.** Un ensemble $E$ est *subpotent* à un ensemble $F$ s'il existe une injection de $E$ dans $F$. On écrit alors $|E| \leqslant |F|$. Si $E$ n'est pas subpotent à $F$, on écrit $|E| \not\leqslant |F|$. ♢

Il est facile de vérifier que cette relation est *transitive*, c'est-à-dire, que si $|E| \leqslant |F|$ et $|F| \leqslant |G|$, alors $|E| \leqslant |G|$. En effet, si les fonctions $f : E \to F$ et $g : F \to G$ sont deux injections, alors leur composition $g \circ f : E \to G$ est une injection témoignant de la relation $|E| \leqslant |G|$. Il est cependant beaucoup moins clair que si $|E| \leqslant |F|$ et $|F| \leqslant |E|$, alors $|E| = |F|$. En déroulant les définitions, la question revient à savoir si, lorsqu'il existe une injection de $E$ dans $F$ et une autre de $F$ dans $E$, il existe une bijection entre $E$ et $F$. Voyons tout de suite que c'est bien le cas : il s'agit du théorème de Cantor–Bernstein.

## 2. Théorème de Cantor–Bernstein

Le cœur de la preuve du théorème de Cantor–Bernstein tient dans le lemme suivant.

**Lemme 2.1.** Si $B \subseteq A$ sont des ensembles, et $f : A \to B$ est une fonction injective, alors il existe une bijection $h : A \to B$. ⋆

Preuve. Le lecteur pourra s'aider de la figure 2.2.

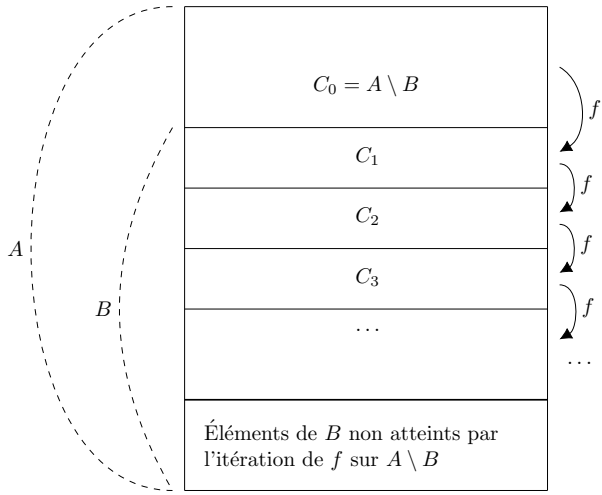



FIGURE 2.2 – *Illustration de la preuve du lemme 2.1*

Soit $C_0, C_1, C_2, \ldots$ la suite définie par récurrence comme suit :

$$C_0 = A \setminus B \quad \text{et} \quad C_{n+1} = f(C_n).$$

On note $C = \bigcup_n C_n$. Un simple raisonnement par récurrence sur $n$ permet de montrer, à l'aide de l'injectivité de $f$, que les ensembles $C_n$ pour $n \in \mathbb{N}$ sont deux à deux disjoints, bien que cela ne soit pas nécessaire dans la preuve. Soit la fonction $h : A \to B$ définie par :

$$h(x) = \begin{cases} f(x) & \text{si } x \in C \\ x & \text{si } x \notin C. \end{cases}$$

Montrons que $h$ est injective. La fonction $f$ étant injective, la fonction $h$ restreinte à $C$ est injective. La fonction $h$ restreinte à $A \setminus C$ est la fonction identité, et est donc également injective. Enfin, l'image de $C = \bigcup_n C_n$ par $h$ est

$$\bigcup_n f(C_n) = \bigcup_n C_{n+1} \subseteq C,$$

et l'image de $A \setminus C$ par $h$ est $A \setminus C$. La fonction $h$ est la réunion de deux fonctions injectives possédant des images disjointes, et est donc injective.

Montrons enfin que $h$ est surjective. Soit $y \in B$ ; si $y \notin C$, alors $h(y) = y$ et donc $y$ a un prédécesseur par $h$. Si $y \in C$, comme $y \notin C_0$, il existe un $n \in \mathbb{N}$ tel que $y \in C_{n+1}$. Par définition de $C_{n+1} = f(C_n)$, il existe un $x \in C_n$ tel que $h(x) = f(x) = y$. ∎

Nous pouvons à présent montrer le théorème annoncé.

**Théorème 2.3 (Cantor-Bernstein)**
*Si $A$ et $B$ sont des ensembles, et $f : A \to B$ et $g : B \to A$ sont des fonctions injectives, alors il existe une bijection entre $A$ et $B$.*

Preuve. Soit $A' = g(B)$. L'application $g \circ f$ est une injection de $A$ dans $A'$. Par le lemme 2.1, il existe donc une bijection $h : A \to A'$. La fonction $g$ étant une bijection entre $B$ et $A'$, la fonction $g^{-1} \circ h : A \to B$ est une bijection. ∎

**Corollaire 2.4**
*Deux ensembles mutuellement subpotents sont équipotents.*

La question de savoir s'il existe des infinis de tailles distinctes, et en particulier avec l'un d'eux strictement plus gros que l'autre, revient donc à savoir si l'on peut trouver deux ensembles $A, B$ pour lesquels $|A| \leqslant |B|$ mais $|B| \not\leqslant |A|$. Nous allons voir que c'est bien le cas.

## 3. Ensembles dénombrables

L'ensemble infini par excellence est bien entendu $\mathbb{N}$, l'ensemble des nombres entiers. On utilise dans ce cas un vocabulaire spécifique.

**Définition 3.1.** Un ensemble infini $A$ est dit *dénombrable* s'il existe une bijection $f : \mathbb{N} \to A$, autrement dit si $A$ et $\mathbb{N}$ sont équipotents. Un ensemble infini $A$ est dit *indénombrable* s'il n'existe pas de bijection $f : \mathbb{N} \to A$. ♢

**Remarque**
Certains auteurs utilisent le terme « dénombrable » pour désigner un ensemble subpotent à $\mathbb{N}$, autrement dit un ensemble fini, ou en bijection avec $\mathbb{N}$. Ils parlent alors d'ensemble *infini dénombrable* pour désigner les ensembles dénombrables tels que nous les avons introduits.

Intuitivement, l'infini des entiers naturels est le plus petit infini, au sens où il est subpotent à tout ensemble infini.

**Proposition 3.2.** L'ensemble $\mathbb{N}$ est subpotent à tout ensemble infini. ⋆

Preuve. Soit $A$ un ensemble infini. Nous allons définir une fonction injective $f : \mathbb{N} \to A$ par récurrence sur $\mathbb{N}$. Soit $f(0)$ un élément quelconque de $A$. Supposons que les valeurs $f(0), \ldots, f(n)$ soient définies. En particulier, $B = \{f(0), \ldots, f(n)\} \subseteq A$ est un ensemble fini tandis que $A$ est infini.

Il existe donc nécessairement un élément dans $A \setminus B$. Soit $f(n+1)$ cet élément. Par construction, $f$ est injective. ∎

Notons qu'en toute généralité, la preuve précédente utilise *l'axiome du choix*, dont nous reparlerons dans la section 9-4, et qui est nécessaire afin de choisir à chaque étape un élément de $A \setminus B$. Toutefois, dans la plupart des cas (comme dans le corollaire suivant), cet axiome n'est pas absolument nécessaire.

**Corollaire 3.3**
*Tout sous-ensemble infini d'un ensemble dénombrable est dénombrable.*

Preuve. Soit $A$ un ensemble dénombrable, et soit $B \subseteq A$ un sous-ensemble infini. L'ensemble $A$ étant équipotent à $\mathbb{N}$, il est donc subpotent à $\mathbb{N}$. Comme $B \subseteq A$, il est subpotent à $A$, donc à $\mathbb{N}$. En outre, par la proposition 3.2, $\mathbb{N}$ est subpotent à $B$. Par le théorème de Cantor-Bernstein (théorème 2.3), $B$ et $\mathbb{N}$ sont équipotents. ∎

**Exercice 3.4. (⋆)** Soit $A$ un ensemble dénombrable, et soit une bijection $f : \mathbb{N} \to A$. Soit enfin $B \subseteq A$ un sous-ensemble infini. Construire directement une bijection de $\mathbb{N}$ vers $B$ qui ne s'appuie pas sur l'axiome du choix. On entend par là que la définition de la fonction doit reposer sur un algorithme explicite, et non sur une procédure abstraite permettant de choisir un élément dans un ensemble non vide, sans que l'on ne sache de quel élément il s'agit. ◇

Introduisons à présent une bijection que nous utiliserons régulièrement dans ce livre, qui est celle couramment utilisée pour attester de la dénombrabilité du produit $\mathbb{N} \times \mathbb{N}$.

**Proposition 3.5.** L'ensemble $\mathbb{N} \times \mathbb{N}$ est dénombrable. ⋆

Preuve. Soit $\alpha : \mathbb{N} \to \mathbb{N}^2$ la fonction telle que les valeurs

$$\alpha(0) = (0,0), \quad \alpha(1) = (1,0), \quad \alpha(2) = (0,1), \ldots$$

énumèrent les couples d'entiers par diagonales successives, comme dans la figure 3.6.

La fonction est injective par construction, et toute paire apparaîtra à une étape de l'énumération. Ainsi, $\alpha$ est une bijection de $\mathbb{N}$ vers $\mathbb{N} \times \mathbb{N}$ témoignant de l'équipotence entre les deux ensembles. ∎

Il est possible de donner une définition analytique de la fonction réciproque de $\alpha$ définie dans la proposition précédente. Il s'agira donc d'une bijection

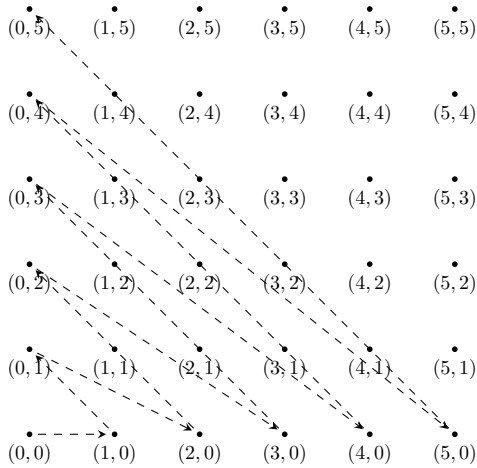



FIGURE 3.6 – *Illustration de la preuve de la proposition 3.5*

de $\mathbb{N}^2$ vers $\mathbb{N}$, que nous appellerons $\alpha_2$ et que le lecteur pourra découvrir à travers l'exercice ci-après.

**Exercice 3.7. (⋆)** Soit $\alpha_2 : \mathbb{N}^2 \to \mathbb{N}$ définie par :

$$\begin{aligned} \alpha_2(x,y) &= y + \sum_{i=0}^{x+y} i \\ &= y + \frac{(x+y+1)(x+y)}{2}. \end{aligned}$$

1. Montrer que $\alpha_2$ est bijective.
2. Montrer que $\alpha_2(a,b) \geqslant a$ et $\alpha_2(a,b) \geqslant b$. ◇

La bijection $\alpha_2$ de l'exercice précédent sera très souvent utilisée dans les développements de ce livre, via la notation suivante.

**Notation**

On note $\langle n, m\rangle$ l'entier sur lequel est envoyé le couple $(n,m)$ via la bijection $\alpha_2$.

Notons que si l'on a une bijection $\alpha_2 : \mathbb{N}^2 \to \mathbb{N}$, on peut définir une bijection $\alpha_3 : \mathbb{N}^3 \to \mathbb{N}$ en prenant simplement $\alpha_3(x,y,z) = \alpha_2\big(x, \alpha_2(y,z)\big)$. On peut continuer ainsi pour définir des bijections de $\mathbb{N}^n$ vers $\mathbb{N}$ pour tout $n \in \mathbb{N}^*$, ce qui conduit à la notation suivante.

**Notation**

On note $\langle x_1, \dots, x_k \rangle$ l'entier sur lequel est envoyé le $k$-uplet $(x_1, \dots, x_k)$ via la bijection de $\mathbb{N}^k$ vers $\mathbb{N}$ décrite ci-dessus.

Profitons de nos bijections fraîchement introduites pour en déduire le corollaire suivant.

**Corollaire 3.8**
*Le produit cartésien de deux ensembles dénombrables est dénombrable.*

Preuve. Soient $A$ et $B$ des ensembles dénombrables et soient $f : A \to \mathbb{N}$ et $g : B \to \mathbb{N}$ des bijections qui en témoignent. La fonction $h : A \times B \to \mathbb{N}$ définie par $h(x, y) = \langle f(x), g(y) \rangle$ est une bijection. ∎

Terminons cette section par trois exercices, permettant de manipuler les concepts vus jusqu'ici. Nous attirons en particulier, l'attention sur le premier d'entre eux que l'on utilisera régulièrement dans les développements à venir.

**Exercice 3.9. (⋆)** Montrer que $\mathbb{Z}$ est un ensemble dénombrable. En déduire que $\mathbb{Z} \times \mathbb{Z}$ est un ensemble dénombrable et enfin que l'ensemble $\mathbb{Q}$ des nombres rationnels — qui s'écrivent sous la forme $p/q$ pour $p, q \in \mathbb{Z}$ avec $q \neq 0$ — est lui aussi dénombrable. ◇

**Exercice 3.10. (⋆)** Soit $f : \mathbb{N} \to A$ une fonction surjective vers un ensemble $A$ infini. Montrer que $A$ est dénombrable. ◇

**Exercice 3.11. (⋆)** Soit $(B_n)_{n \in \mathbb{N}}$ une suite d'ensembles dénombrables avec des bijections respectives $f_n$ avec $\mathbb{N}$. Montrer que $B = \bigcup_n B_n$ est un ensemble dénombrable.

Attention : si l'on ne dispose pas des bijections $(f_n)_{n \in \mathbb{N}}$, on peut toujours montrer que $B$ est dénombrable, mais il faut utiliser l'*axiome du choix*, afin de choisir uniformément pour chaque $B_n$ une de ses bijections avec $\mathbb{N}$. Nous en reparlerons dans la section 9-4. ◇

# 4. Argument diagonal de Cantor

Attelons-nous à présent au théorème annoncé au début de ce chapitre : il existe des infinis plus gros que d'autres, et en particulier un infini plus gros que celui des entiers. Le théorème suivant utilise le fameux argument diagonal de Cantor, qui sera repris à de très nombreuses occasions et sous des formes variées tout au long de ce livre.

| $R$ | $=$ | $0,$ | $9-x_{00}$ | $9-x_{11}$ | $9-x_{22}$ | $9-x_{33}$ | $9-x_{44}$ | $9-x_{55}$ | $9-x_{66}$ |
|---|---|---|---|---|---|---|---|---|---|
| $f(0)$ | $=$ | $N_0,$ | $\boldsymbol{x_{00}}$ | $x_{01}$ | $x_{02}$ | $x_{03}$ | $x_{04}$ | $x_{05}$ | $x_{06}$ |
| $f(1)$ | $=$ | $N_1,$ | $x_{10}$ | $\boldsymbol{x_{11}}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{16}$ |
| $f(2)$ | $=$ | $N_2,$ | $x_{20}$ | $x_{21}$ | $\boldsymbol{x_{22}}$ | $x_{23}$ | $x_{24}$ | $x_{25}$ | $x_{16}$ |
| $f(3)$ | $=$ | $N_3,$ | $x_{30}$ | $x_{31}$ | $x_{32}$ | $\boldsymbol{x_{33}}$ | $x_{34}$ | $x_{35}$ | $x_{36}$ |
| $f(4)$ | $=$ | $N_4,$ | $x_{40}$ | $x_{41}$ | $x_{42}$ | $x_{43}$ | $\boldsymbol{x_{44}}$ | $x_{45}$ | $x_{46}$ |
| $f(5)$ | $=$ | $N_5,$ | $x_{50}$ | $x_{51}$ | $x_{52}$ | $x_{53}$ | $x_{54}$ | $\boldsymbol{x_{55}}$ | $x_{56}$ |
| $f(6)$ | $=$ | $N_6,$ | $x_{60}$ | $x_{61}$ | $x_{62}$ | $x_{63}$ | $x_{64}$ | $x_{65}$ | $\boldsymbol{x_{66}}$ |
| $\ldots$ | | | | | | | | | |

FIGURE 4.2 – *Illustration de l'argument diagonal de Cantor. On construit notre réel $R$ à partir de la diagonale du tableau, la décimale $9-x_{ii}$ étant nécessairement différente de $x_{ii}$. Notons que le nombre $R$ illustré ici n'est pas exactement celui décrit dans la preuve, mais le résultat est le même : l'élément $R$ n'est pas dans l'image de $f$.*

**Théorème 4.1 (Cantor)**
*L'ensemble des nombres réels est indénombrable.*

PREUVE. Le lecteur peut s'aider de la figure 4.2, qui illustre l'argument de la preuve. On raisonne par l'absurde. Supposons au contraire qu'il existe une bijection $f : \mathbb{N} \to \mathbb{R}$. Nous allons construire un nombre réel $R \in \mathbb{R}$ qui n'est pas dans l'image de $f$. On définit simplement $R$ de la manière suivante : la partie entière de $R$ est 0, et pour tout $n$, si la $n$-ième décimale de $f(n)$ est différente de 0, alors la $n$-ième décimale de $R$ est égale à 0. À l'inverse, si la $n$-ième décimale de $f(n)$ est égale à 0, alors la $n$-ième décimale de $R$ est égale à 1.

Il est clair que pour tout entier $n$, notre nombre réel $R$ ne peut être égal à $f(n)$, car la $n$-ième décimale de $R$ est différente de la $n$-ième décimale de $f(n)$. ∎

Que nous dit le théorème précédent ? Qu'il y a « plus » de nombres réels que de nombres entiers. Ces ensembles de nombres sont tous les deux infinis, mais l'infini des réels est « plus grand » que celui des entiers. En effet, quand on essaye de faire correspondre un nombre entier à chaque nombre réel, on s'aperçoit qu'il y a toujours des réels « en trop », qui ne correspondent à aucun entier. On a donc $|\mathbb{R}| \not\leqslant |\mathbb{N}|$. Notons que l'on a en revanche $|\mathbb{N}| \leqslant |\mathbb{R}|$, via l'injection identité. On pourra dans ce cas écrire $|\mathbb{N}| < |\mathbb{R}|$, signifiant qu'il y a une injection de $\mathbb{N}$ dans $\mathbb{R}$, mais pas d'injection de $\mathbb{R}$ dans $\mathbb{N}$.

Y a-t-il un infini plus grand que celui des réels ? Cantor a également répondu à cette question, par un argument similaire : pour tout ensemble $A$, il existe un ensemble $B$ tel que $|A| < |B|$.

**Théorème 4.3 (Cantor)**
*Pour tout ensemble $A$, l'ensemble $\mathcal{P}(A)$ constitué des sous-ensembles de $A$ est tel que $|A| < |\mathcal{P}(A)|$.*

Preuve. L'ensemble $A$ est subpotent à $\mathcal{P}(A)$, en considérant l'injection qui à tout élément $x \in A$ associe le singleton $\{x\}$. Ainsi, $|A| \leqslant |\mathcal{P}(A)|$. Supposons à présent par l'absurde que $|\mathcal{P}(A)| = |A|$, c'est-à-dire supposons qu'il existe une bijection $f : A \to \mathcal{P}(A)$. Considérons l'ensemble

$$B = \{x \in A \ : \ x \notin f(x)\},$$

et soit $y$ tel que $f(y) = B$. Alors, $y \in B$ ssi $y \notin f(y)$, autrement dit ssi $y \notin B$, ce qui est une contradiction. ∎

On peut légitimement se demander si la cardinalité des ensembles est toujours comparable : étant donné deux ensembles $A, B$, existe-t-il toujours une injection de l'un dans l'autre ? La réponse à cette question dépend là encore de *l'axiome du choix*. Nous en reparlerons dans la section 9-4 ainsi que dans la section 27-4.1.

## 5. Réels non calculables

Le théorème de Cantor va nous fournir notre premier argument de l'existence de nombres réels incalculables. Nous utilisons pour la proposition et le corollaire suivants les notions pour le moment informelles de « programme informatique » et de « réel calculable ». Elles seront toutes les deux précisément définies plus loin dans ce livre ; considérons pour l'instant qu'un réel est calculable s'il existe un programme informatique qui énumère dans l'ordre la liste infinie de ses décimales.

**Proposition 5.1.** L'ensemble des programmes informatiques est dénombrable. ⋆

Preuve. Un programme informatique (écrit dans quelque langage de programmation que ce soit) se présente sous la forme d'une suite finie de caractères, où chaque caractère appartient à un alphabet fini (disons l'ensemble des caractères de la table ascii). Montrons qu'il existe une bijection de $\mathbb{N}$ vers l'ensemble des suites finies de caractères ascii. Pour cela, nous définissons une liste infinie de toutes les suites finies de caractères ascii : on liste d'abord toutes les suites comportant exactement un caractère ascii,

puis toutes les suites comportant deux caractères ascii, puis toutes celles en comportant trois, etc. Il est clair que n'importe quelle suite finie de caractères ascii apparaît quelque part dans notre liste infinie.

Il suffit à présent de retirer de notre liste les suites finies ne correspondant pas à un programme valide. On définit alors l'élément $f(n)$ comme étant le $n$-ième élément de la liste résultant de cette opération. Il est clair que $f$ est une bijection. En particulier, l'ensemble des programmes informatiques est dénombrable. ∎

**Corollaire 5.2**
*Il existe des nombres réels qui ne sont pas calculables.*

Preuve. Soit $P$ l'ensemble des programmes informatiques. Supposons que tout réel est calculé par un élément de $P$. Soit $p_1$ le premier programme informatique calculant un réel. Ici « premier » veut dire premier selon l'ordre obtenu via la bijection entre les programmes informatiques et $\mathbb{N}$. Supposons que $p_1, \ldots, p_n$ soient définis et calculent tous un réel différent. Soit $p_{n+1}$ le premier programme informatique calculant un réel différent de ceux calculés par $p_1, \ldots, p_n$. On définit par récurrence de cette manière la suite $(p_n)_{n\in\mathbb{N}}$.

On obtient alors une bijection entre $\mathbb{R}$ et un sous-ensemble infini de $P$. Comme $P$ est dénombrable, ce sous-ensemble infini l'est également, ce qui donne une bijection entre $\mathbb{N}$ et $\mathbb{R}$, et dès lors une contradiction. ∎

La preuve du corollaire 5.2 est non constructive : on montre l'existence de nombres incalculables sans en donner d'exemple précis. Ce sera évidemment fait à maintes reprises dans la suite de cet ouvrage, à travers une étude détaillée de différents types de nombres incalculables. Avant de nous y atteler, mentionnons deux ou trois notions importantes concernant l'étude de l'infini menée par Cantor suite à sa découverte.

## 6. Espace de Cantor

Cantor a montré qu'il n'y avait pas de plus grand infini. Parmi tous les infinis possibles, le plus petit est « le dénombrable ». Un autre infini nous intéresse également, celui des nombres réels :

**Définition 6.1.** Un ensemble $A$ est dit *avoir la puissance du continu* si $|A| = |\mathbb{R}|$. ♢

La puissance du continu caractérise donc l'infini des réels. Cantor conjectura qu'il n'y avait pas d'infini strictement compris entre $|\mathbb{N}|$ et $|\mathbb{R}|$, mais

sans réussir à le démontrer. Sa conjecture connue sous le nom *d'hypothèse du continu* sera considérée pendant près d'un siècle comme une des plus importantes questions mathématiques. Gödel montrera en 1938 [74] qu'il n'est pas possible de démontrer que l'hypothèse du continu est fausse, et Cohen mettra un point final à la question en 1963 [37] en montrant qu'il n'est pas possible non plus de montrer que l'hypothèse du continu est vraie : il s'agit d'une question indépendante du reste des mathématiques, que l'on peut considérer vraie ou fausse sans introduire de contradiction. Nous en rediscuterons plus en détail dans la section 9-4.

Parmi les ensembles ayant la puissance du continu, on prêtera une attention particulière à l'ensemble des suites infinies de 0 et de 1, qui constituera l'essentiel de notre terrain de jeu tout au long de cet ouvrage. Par suite infinie de 0 et de 1, on entend suite indexée par les entiers, c'est-à-dire des suites de la forme $x_0x_1x_2x_3\ldots$ où chaque $x_i \in \{0,1\}$.

**Définition 6.2.** On appelle *espace de Cantor* l'ensemble des suites infinies de 0 et de 1, on le note $2^{\mathbb{N}}$, et ses éléments seront dénotés via des lettres majuscules, en général $A, B, C, X, Y, Z$. ♢

L'espace de Cantor a la puissance du continu.

**Proposition 6.3.** On a : $|2^{\mathbb{N}}| = |[0,1]| = |\mathbb{R}|$. ⋆

Preuve. Montrons d'abord $|[0,1]| = |\mathbb{R}|$. La fonction identité est une injection de $[0,1]$ dans $\mathbb{R}$. On vérifie sans peine que la fonction

$$f(x) = \begin{cases} 1/2 + 1/(x+2) & \text{si } x \geqslant 0 \\ 1/2 - 1/(|x|+2) & \text{si } x < 0 \end{cases}$$

est une injection de $\mathbb{R}$ dans $[0,1]$. Par le théorème de Cantor-Bernstein (voir le théorème 2.3), on a donc $|[0,1]| = |\mathbb{R}|$.

Montrons à présent $|2^{\mathbb{N}}| = |[0,1]|$. Pour définir une injection de $2^{\mathbb{N}}$ dans $[0,1]$, il convient d'être prudent : certains nombres réels ont deux développements binaires possibles, ainsi $1.00000\cdots = 0.11111\ldots$ — où $0.11111\ldots$ est le nombre décimal $0.99999\ldots$ écrit en binaire. Il en va de même pour tout nombre réel dont le développement binaire se termine par une infinité de 0 consécutifs : celui-ci a alors un développement binaire équivalent qui se termine par une infinité de 1 consécutifs. On définira donc l'injection $f : 2^{\mathbb{N}} \to \mathbb{R}$ qui à $X$ associe le nombre réel de $[0,1]$ dont le développement *ternaire*, c'est-à-dire en base 3, est constitué des bits de $X$. L'usage de la représentation ternaire permet de contourner ces problèmes d'égalités et de rendre ainsi la fonction $f$ injective. L'injection $g : [0,1] \to 2^{\mathbb{N}}$ est définie en associant à tout réel $r \in [0,1]$ son développement binaire $R$.

Lorsqu'il existe plusieurs représentations du même nombre réel, on choisira par convention celle se terminant par une infinité de 0. Par le théorème de Cantor-Bernstein, on a donc $|2^{\mathbb{N}}| = |[0,1]|$. ∎

**Notation**

Étant donné $X \in 2^{\mathbb{N}}$ et $n \in \mathbb{N}$, on note $X(n)$ le $n$-ième élément de la suite $X$ que l'on appellera aussi le $n$-ième *bit* de $X$. Ainsi, si la suite $X$ s'écrit $x_0x_1x_2\ldots$, alors $X(0) = x_0$, $X(1) = x_1$, $X(2) = x_2$, ...

Le fait que certains réels aient deux représentations binaires possibles font de $|2^{\mathbb{N}}|$ et $|[0,1]|$ des espaces topologiquement différents. Par exemple, pour tous $x, y \in \mathbb{R}$ avec $x < y$, il existe un réel $z$ strictement entre les deux. En revanche, si l'on munit $2^{\mathbb{N}}$ de l'ordre lexicographique $<_{lex}$ défini par

$$X <_{lex} Y \quad \text{si} \quad X(n) < Y(n),$$

où $n$ est le plus petit entier tel que $X(n) \neq Y(n)$, alors les suites infinies

$$X = 0111111\ldots \quad \text{et} \quad Y = 100000\ldots$$

n'ont aucun élément strictement entre elles pour cet ordre.

Cette différence est anecdotique du point de vue de la complexité calculatoire des éléments, et il nous arrivera parfois de parler de réel ou de suites binaires infinies indistinctement. La différence a toutefois son importance pour le développement de la calculabilité dans d'autres espaces topologiques que $2^{\mathbb{N}}$. Nous en discuterons brièvement dans la section 22-4.2.

Voyons à présent qu'il existe en revanche une bijection très naturelle entre l'ensemble $\mathcal{P}(\mathbb{N})$ — l'ensemble des parties de $\mathbb{N}$ — et l'espace de Cantor. Ainsi, $\mathcal{P}(\mathbb{N})$ a la puissance du continu.

**Proposition 6.4.** On a : $|2^{\mathbb{N}}| = |\mathcal{P}(\mathbb{N})|$. ⋆

Preuve. Soit $f : 2^{\mathbb{N}} \to \mathcal{P}(\mathbb{N})$ la fonction qui à $X \in 2^{\mathbb{N}}$ associe l'ensemble $Y = \{n \in \mathbb{N} : X(n) = 1\}$. La fonction $f$ est clairement une bijection. ∎

La bijection entre l'ensemble $\mathcal{P}(\mathbb{N})$ des parties de $\mathbb{N}$ et l'espace de Cantor $2^{\mathbb{N}}$ est tellement élémentaire que l'on peut considérer $\mathcal{P}(\mathbb{N})$ et $2^{\mathbb{N}}$ comme deux représentations du même concept mathématique. Dans la suite, nous parlerons indistinctement de sous-ensemble de $\mathbb{N}$ ou de suite infinie de 0 et de 1, et utiliserons la même notation $2^{\mathbb{N}}$ pour désigner l'ensemble de ces éléments. Ainsi, étant donné $X \in 2^{\mathbb{N}}$ vu comme une suite, on aura $X(n) = 1$ ssi $n$ appartient à $X$ vu comme un ensemble.

**Digression**

Le nom « espace de Cantor » vient sans doute du concept éponyme *d'ensemble triadique de Cantor*. On définit $A_0 = [0, 1]$, puis $A_1$ est $A_0$ moins son tiers central :

$$A_1 = [0, 1/3] \cup [2/3, 1].$$

Puis $A_2$ est $A_1$ moins le tiers central de chacun de ses intervalles :

$$A_2 = [0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1].$$

De manière générale, pour passer de $A_n$ à $A_{n+1}$, on enlève le tiers central de chacun des intervalles de $A_n$. L'ensemble triadique de Cantor est le résultat à la limite, de l'application de cette opération, c'est-à-dire l'ensemble $\bigcap_{n\in\mathbb{N}} A_n$.

L'espace de Cantor $2^{\mathbb{N}}$ défini plus haut, est topologiquement équivalent à l'ensemble triadique de Cantor. En particulier, chaque point de $\bigcap_{n\in\mathbb{N}} A_n$ peut être décrit comme une suite de 0 et de 1 de la manière suivante : le $n$-ième bit d'un point correspond à déterminer s'il est à droite ou à gauche du $n$-ième tiers que l'on enlève de l'intervalle courant. On peut également voir l'espace de Cantor comme l'ensemble des réels de $[0, 1]$ dont la représentation ternaire évite le chiffre 1.

# Première partie

# Calculabilité classique

Chapitre 3

# Fondements de la calculabilité

Notre objectif est de mener une étude mathématique de la calculabilité. Pour ce faire, il est d'usage de définir mathématiquement ce que l'on entend par *fonction calculable*. Cette quête d'une définition formelle capturant ce concept épistémologique a constitué la genèse de la calculabilité, et abouti à ce que l'on appelle de nos jours la *thèse de Church-Turing*. Cette thèse énonce que tout processus calculable peut être exécuté avec une *machine de Turing*, un modèle de calcul imaginé par Alan Turing en 1936 et qui peut être considéré comme un précurseur des ordinateurs modernes. D'autres approches que celle des machines de Turing permettent de capturer la notion de fonction calculable, parmi lesquelles nous citerons les fonctions générales récursives et le $\lambda$-calcul. Ces différents modèles seront présentés plus en détail dans l'interlude sur la thèse de Church-Turing (voir le chapitre 6), et nous adopterons pour ce chapitre une approche moins formelle.

## 1. Fonctions calculables

En calculabilité, le formalisme des machines de Turing tend à servir de modèle de référence, non seulement pour des raisons historiques — Turing fut le premier à avoir su convaincre la communauté que son modèle capturait tous les processus calculables — mais également parce que ce modèle met en relief la notion d'étape atomique de calcul, ce qui ouvre la porte à la théorie de la complexité. Il est de coutume de débuter l'étude de la calculabilité par celle de ses modèles de calcul, et de prouver leur équivalence, afin de se convaincre de la robustesse de la notion de fonction calculable

et du bien-fondé des définitions. Il s'agit d'un développement assez long et fastidieux. Aussi est-il facile d'être rebuté par cette étape à l'issue de laquelle on croit trop facilement — à tort — que la calculabilité se résume à de complexes et rébarbatives techniques de codage.

Nous avons donc fait le choix de rompre avec la tradition, et reporter la définition et l'étude mathématique des modèles de calcul au chapitre 6, afin de faciliter le premier contact avec la calculabilité, et d'accéder plus directement à ses concepts fondamentaux. L'avènement des ordinateurs et la démocratisation de l'enseignement de la programmation ont ancré durablement la notion d'algorithme dans la culture scientifique. Nous adopterons donc la définition informelle suivante.

**Définition 1.1.** Une fonction $f : \mathbb{N} \to \mathbb{N}$ est *calculable* si elle peut être définie par un algorithme, ou autrement dit programmée dans un langage de programmation moderne. ♢

Il s'ensuit de notre choix pédagogique que les preuves de nos premiers théorèmes feront largement appel à l'intuition des propriétés que l'on attend d'une fonction calculable. Il s'agit cependant de théorèmes à part entière, dans le sens où il est possible de les prouver à partir des définitions formelles du chapitre 6.

**Entiers en informatique**

Nous nous intéressons essentiellement aux fonctions de $\mathbb{N}$ dans $\mathbb{N}$. Dans la plupart des langages de programmation standard, les entiers sont bornés. Pour notre définition théorique, il est important de prendre en compte *tous les entiers*.

**Algorithmes**. Afin de s'accorder sur ce que l'on entend par un langage de programmation moderne ou algorithme, listons-en les principaux aspects, qui seront repris formellement dans la section 6-3 pour notre définition des programmes structurés sur le modèle des machines à registres.

(1) Le langage doit pouvoir manipuler les entiers, à l'aide de constantes, de variables entières, et des opérations arithmétiques usuelles, à savoir l'addition, la soustraction, la multiplication, et la division entière. Le lecteur pourra y ajouter d'autres types, comme les chaînes de caractères ou les nombres à virgule flottante, sans que cela ne change la puissance de calcul.

(2) Le langage doit pouvoir manipuler des expressions booléennes, et effectuer des opérations de comparaison sur les entiers.

(3) Le langage doit contenir les structures de contrôle usuelles, à savoir des instructions conditionnelles de type « `if` ... `then` ... `else` ... » et

des boucles « `for` » et « `while` », qui se répètent tant qu'une certaine condition est vraie.

(4) Nous supposerons que la mémoire de la machine n'est pas bornée, et qu'elle peut en utiliser autant que nécessaire (notamment pour lire son entrée, qui peut être un entier arbitrairement grand).

**Types de données**

Le point (1) insiste sur le fait que l'ajout d'autres types de données que le type entier est superflu. C'est vrai à condition bien sûr de considérer que nos entiers peuvent être arbitrairement grands (pour encoder par exemple de grandes chaînes de caractères), ce qui sera par convention toujours le cas. Le lecteur pourra trouver un exemple d'encodage de tableaux par des entiers dans la proposition 6-3.26.

**Mémoire non bornée**

On peut être surpris par le point (4) ci-dessus, qui autorise une mémoire non bornée. Insistons sur le fait que l'on ne s'autorise pas une mémoire infinie pour autant : un calcul qui se termine n'a effectué qu'un nombre fini d'opérations et n'a donc pu utiliser qu'une quantité finie de mémoire. Simplement, on ne s'occupe pas en calculabilité de la complexité en espace des algorithmes.

**Exemples**. Avant de commencer l'étude formelle des fonctions calculables et de leurs propriétés, listons quelques exemples de fonctions calculables, afin de commencer à nous former une intuition.

(1) Les opérations arithmétiques usuelles sont calculables. En particulier, l'addition, la multiplication, la soustraction et la division entière sont calculables.

(2) La fonction qui à $n$ associe le $n$-ième nombre premier est calculable, de même que la décomposition d'un nombre en ses facteurs premiers.

(3) La fonction qui, prenant en paramètre une liste de positions de villes, renvoie un des chemins les plus courts passant par toutes ces villes une seule fois, est calculable [1].

Inversement, il existe comme nous allons le voir de nombreuses fonctions non calculables. Cependant, tandis qu'il suffit de donner un algorithme pour montrer qu'une fonction est calculable, montrer qu'une fonction n'est pas calculable demande souvent un raisonnement plus élaboré, car il ne suffit pas que la fonction n'ait pas d'algorithme connu ; il faut montrer qu'il est

1. Il s'agit du problème bien connu du *voyageur de commerce*.

théoriquement impossible de la programmer. Chacun des énoncés suivants est donc un théorème à part entière.

(1) La fonction qui prend en entrée un programme informatique (codé par un entier ou une chaîne de caractères), et décide si son exécution va un jour s'arrêter, n'est pas calculable (voir le théorème 7.8).

(2) La fonction qui prend en entrée une formule dans le langage de l'arithmétique (par exemple, « $\forall x\, \forall y\, \forall z\, x^3+y^3 \neq z^3$ »), et renvoie 1 s'il existe une démonstration mathématique de cette formule dans le système axiomatique de l'arithmétique, et 0 sinon, n'est pas calculable (voir le théorème 9-3.9).

(3) La fonction qui prend en entrée une équation diophantienne, i.e. une équation à coefficients entiers — par exemple $3x^2 + 2xy + 4y^2 = 0$ — et décide si cette équation admet une solution entière, n'est pas calculable (voir le théorème 12-1.2).

**Fonctions partielles**. Comme expliqué dans l'encadrement « Entiers en informatique » ci-dessus, on se restreindra en général aux programmes prenant en paramètre un entier, et retournant un autre entier si le calcul se termine. Il est essentiel de noter que les fonctions engendrées par les programmes informatiques sont *partielles*, au sens où le calcul peut ne jamais s'arrêter sur certaines de leurs entrées. Cette partialité est due aux structures de contrôle de type « `while` » dont la condition de terminaison peut n'être jamais satisfaite, comme le montre l'exemple suivant, qui calcule — de la pire manière possible — la racine carrée d'un entier :

```
function Racine(n) {
    r = 0;
    while(r*r ≠ n) {
      r = r+1;
    }
    return r;
}
```

Si $n$ n'est pas le carré d'un nombre entier, la boucle `while` va s'exécuter à l'infini, et le programme ne renverra jamais de valeur. Lorsque le programme ne s'arrête pas sur une entrée $n$, on considère que la fonction de $\mathbb{N}$ dans $\mathbb{N}$ qui lui est associée n'est pas définie sur $n$. Le domaine de définition de la fonction partielle associée à un programme est donc l'ensemble des entrées sur lesquelles il s'arrête. Les fonctions définies par des programmes sont appelées *fonctions partielles calculables*. Lorsque le programme s'arrête sur toutes ses entrées, la fonction engendrée est alors appelée *fonction totale calculable*, ou tout simplement *fonction calculable*.

**Notation**

Si $f$ et $g$ sont deux fonctions partielles, on notera $f(x) = g(x)$ pour signifier soit que $f$ et $g$ sont toutes les deux définies et renvoient la même valeur sur $x$, soit que ni $f$ ni $g$ ne sont définies sur $x$.

**Codes.** Dans notre étude mathématique, nous représenterons les programmes informatiques par des entiers, en supposant une fonction de codage fixée. Par *programmes informatiques*, il faut comprendre ici une suite finie de caractères — supposée avoir du sens dans un langage de programmation choisi au préalable. Concrètement, on dira qu'un entier $e$ *code* pour un programme $P$ si $e$ est l'entier codé par la représentation binaire de la chaîne de caractères correspondant au programme $P$. En particulier, ce codage est injectif et intuitivement calculable. Notons que l'on pourrait imaginer beaucoup d'autres codages possibles. Nous en verrons un autre de manière détaillée dans la preuve du théorème 6-3.27. En attendant, celui-ci conviendra parfaitement.

**Notation**

On note $\Phi_e : \mathbb{N} \to \mathbb{N}$ la fonction partielle définie par le programme informatique de code $e$.

On supposera que tout entier $e$ code pour un programme valide. En pratique, il existe dans tous les langages de programmation des chaînes de caractères correspondant à des programmes mal formés. Certains entiers $e$ codent pour des suites de caractères inintelligibles. Si c'est le cas, on peut alors s'en rendre compte (cela correspond à avoir une erreur de syntaxe quand on essaye de compiler un programme) et considérer que $e$ correspond alors à un programme qui ne s'arrête pas.

**Notation**

Soient $\Phi_e : \mathbb{N} \to \mathbb{N}$ une fonction partielle calculable et $x \in \mathbb{N}$ un entier. On écrira $\Phi_e(x)\downarrow$ si le programme codé par $e$ s'arrête sur l'entrée $x$ (nécessairement après un nombre fini d'étape de calcul) et renvoie un résultat entier. Dans le cas inverse, on écrira $\Phi_e(x)\uparrow$.

Le domaine de définition de la fonction partielle calculable $\Phi_e : \mathbb{N} \to \mathbb{N}$ est donc $\{x \in \mathbb{N} : \Phi_e(x)\downarrow\}$. Si $\Phi_e(x)\downarrow$, on écrira parfois $\Phi_e(x)\downarrow = y$ pour signifier que le programme de code $e$ s'arrête sur l'entrée $x$ et renvoie l'entier $y$. À l'inverse, on écrira parfois $\Phi_e(x)\uparrow \neq y$ pour signifier l'opposé, c'est-à-dire $\Phi_e(x)\uparrow \ \vee \Phi_e(x)\downarrow \neq y$.

**Temps de calcul**. Tout langage de programmation vient avec une notion d'étape d'exécution et de temps de calcul. Une étape d'exécution est une opération atomique du langage, indécomposable en sous-étapes. Elle correspond à une instruction élémentaire du langage. La notion d'étape d'exécution induit celle de temps de calcul, qui se définit par le nombre d'étapes d'exécution réalisées depuis le lancement du calcul. Lorsqu'un programme s'arrête sur une entrée, son temps de calcul est fini.

**Notation**

Soient $\Phi_e : \mathbb{N} \to \mathbb{N}$ une fonction partielle calculable et $x, t \in \mathbb{N}$ deux entiers. On écrira $\Phi_e(x)[t]\downarrow$ si le programme codé par $e$ s'arrête *avant* $t$ *étapes de calcul*. Dans le cas inverse, on écrira $\Phi_e(x)[t]\uparrow$.

Notons que $\Phi_e(x)\downarrow$ ssi il existe un temps de calcul $t$ tel que $\Phi_e(x)[t]\downarrow$. En outre, si $\Phi_e(x)[t]\downarrow$, alors $\Phi_e(x)[s]\downarrow$ pour tout $s \geqslant t$.

**Remarque**

On sera amené à manipuler des fonctions calculables à plusieurs paramètres. Pour un entier $n \in \mathbb{N}^*$ fixé, on désignera comme ci-dessus par $\Phi_e : \mathbb{N}^n \to \mathbb{N}$ la fonction partielle à $n$ paramètres entiers codée par $e$ et l'on écrira $\Phi_e(x_1, \ldots, x_n)\downarrow = y$ si le programme de code $e$ s'arrête sur les entrées $x_1, \ldots, x_n$ et renvoie l'entier $y$.

## 2. Ensembles calculables

L'adjectif « calculable » s'applique naturellement aux fonctions ; en effet, comme décrit plus haut, tout entier $e$ code pour un programme auquel correspond une fonction partielle calculable. Le plus souvent cependant, quand on parle de fonctions calculables, on considérera qu'il s'agit de fonctions totales.

**Définition 2.1.** Soit $n \in \mathbb{N}^*$. Une fonction $f : \mathbb{N}^n \to \mathbb{N}$ est *calculable* s'il existe un code de programme $e$ tel que, pour tout $(x_1, \ldots, x_n)$, on a $\Phi_e(x_1, \ldots, x_n)\downarrow = f(x_1, \ldots, x_n)$. ♢

La calculabilité peut également être utilisée pour mesurer la complexité descriptive des objets mathématiques dénombrables, et en particulier celle des ensembles d'entiers. Intuitivement, un ensemble d'entiers $E \subseteq \mathbb{N}$ est calculable s'il peut être décrit par un processus calculable. Un ensemble étant totalement spécifié par ses éléments, il est calculable si sa fonction caractéristique est calculable.

**Définition 2.2.** Soit $n \in \mathbb{N}^*$. Un ensemble $A \subseteq \mathbb{N}^n$ est calculable s'il existe un code de programme $e$ tel que, pour tous $x_1, \ldots, x_n$, on a

- ▷ $\Phi_e(x_1, \ldots, x_n)\downarrow = 1$ ssi $(x_1, \ldots, x_n) \in A$.
- ▷ $\Phi_e(x_1, \ldots, x_n)\downarrow = 0$ ssi $(x_1, \ldots, x_n) \notin A$. ♢

On appellera parfois *prédicats* les sous-ensembles de $\mathbb{N}^n$, en utilisant alors la notation $A(x_1, \ldots, x_n)$ pour signifier $(x_1, \ldots, x_n) \in A$. Le terme de prédicat vient de la vision de $A \subseteq \mathbb{N}^n$ non comme un ensemble, mais comme une propriété des $n$-uplets d'entiers. Ainsi, $A(x_1, \ldots, x_n)$ signifie que le $n$-uplet $(x_1, \ldots, x_n)$ a la propriété $A$. À l'inverse, $\neg A(x_1, \ldots, x_n)$ signifie que le $n$-uplet $(x_1, \ldots, x_n)$ n'a pas la propriété $A$.

**Exercice 2.3.** Montrer que la bijection de couplage définie dans la proposition 2-3.5 et l'exercice 2-3.7, qui à $(a, b) \in \mathbb{N}^2$ associe $\langle a, b\rangle \in \mathbb{N}$, est calculable. Montrer que les fonctions inverses $\pi_0, \pi_1$ telles que $a = \pi_0(\langle a, b\rangle)$ et $b = \pi_1(\langle a, b\rangle)$ sont elles aussi calculables. ◇

**Exercice 2.4.** Soit $A \subseteq \mathbb{N}^2$ un prédicat calculable. Montrer que les ensembles :

(1) $\{(x, y) \in \mathbb{N}^2 \ : \ \forall z < y \ (x, z) \in A\}$

(2) $\{(x, y) \in \mathbb{N}^2 \ : \ \exists z < y \ (x, z) \in A\}$

sont eux aussi calculables. ◇

**Ensemble récursif**

Historiquement, les ensembles calculables étaient appelés *récursifs* en raison du paradigme de calcul des fonctions générales récursives dans lequel les définitions par récurrence jouent un rôle prépondérant (voir le chapitre 6). Peu à peu, avec l'amélioration de notre compréhension de la notion de calcul, la terminologie du domaine a évolué. On peut cependant régulièrement voir les termes de *théorie de la récursion* et d'*ensemble récursif* pour parler de calculabilité et d'ensemble calculable.

# 3. Programme universel

Le modèle de calcul imaginé par Turing en 1936 consiste à définir une machine pour chaque fonction calculable. Si l'on compare une machine de Turing à un dispositif physique, cela consiste, pour chaque tâche que l'on veut accomplir, à créer un robot exécutant cette tâche spécifique.

Nous allons maintenant énoncer un premier théorème fondamental de la calculabilité et prouvé par Turing dans son article original : l'existence

d'une machine de Turing *universelle*, capable de simuler toutes les autres machines de Turing. Ce travail est parfois considéré comme précurseur de l'architecture de von Neumann[2], dont l'un des aspects est le stockage des programmes dans la mémoire. En informatique moderne, ce théorème peut être vu comme énonçant l'existence d'un programme informatique *universel*, permettant de simuler tous les autres programmes informatiques.

**Théorème 3.1**
*Soit $n \in \mathbb{N}^*$. Il existe un code $e$ de programme informatique pour lequel $\Phi_e : \mathbb{N}^{n+1} \to \mathbb{N}$ est tel, que pour tous $x_1, \ldots, x_n$, on a*

- ▷ $\Phi_e(a, x_1, \ldots, x_n)\uparrow$ *ssi* $\Phi_a(x_1, \ldots, x_n)\uparrow$ ;
- ▷ $\Phi_e(a, x_1, \ldots, x_n)\downarrow = y$ *ssi* $\Phi_a(x_1, \ldots, x_n)\downarrow = y$.

En pratique, un programme universel existe très concrètement dans beaucoup de langages. Par exemple, en Java, il s'agit simplement de la machine virtuelle qui compile et exécute n'importe quel programme écrit en Java. Pour les langages n'utilisant pas de machine virtuelle, un programme universel sera simplement la donnée d'un interpréteur qui décompose le programme qu'il reçoit en une série d'instructions qu'il exécute pas à pas. Si un tel programme est bien entendu complexe à concevoir, il ne devrait faire aucun doute pour le programmeur qu'il s'agit de quelque chose de tout à fait possible : il s'agit simplement d'une machine virtuelle. Le lecteur pourra consulter la preuve du théorème 6-3.27, qui démontre l'existence d'un programme universel pour le modèle de calcul spécifique des machines à registre.

L'existence d'un tel programme nous permet de définir des fonctions qui font des manipulations sur des codes avant de les exécuter, ou exécutent dynamiquement des codes passés en paramètre. Par exemple,

$$f(x, y) = \Phi_{x+1}(y + 2)$$

est une définition valide, car elle est équivalente à $f(x, y) = \Phi_e(x+1, y+2)$, où $\Phi_e$ est le programme universel.

## 4. Théorème SMN

Le théorème SMN est notre premier théorème sur la manipulation des codes de programmes informatiques. Il ne s'agit pas de quelque chose de conceptuellement bien difficile. Soit $\Phi_e : \mathbb{N}^{m+n} \to \mathbb{N}$ la fonction de code $e$. Alors, étant donné $x_1, \ldots, x_m$, on peut transformer de manière calculable

2. Qui est à peu de choses près toujours celle utilisée de nos jours.

notre code $e$ en un code $a$ tel que le calcul $\Phi_a(y_1, \ldots, y_n)$ donne le même résultat que le calcul $\Phi_e(x_1, \ldots, x_m, y_1, \ldots, y_n)$. Le point important est que la transformation de $e$ en $a$ est calculable en fonction de $e$ et de $x_1, \ldots, x_m$.

**Théorème 4.1 (Théorème SMN)**
*Pour tous $n, m \in \mathbb{N}^*$, il existe une fonction totale calculable*

$$S_n^m : \mathbb{N}^{m+1} \to \mathbb{N},$$

*telle que pour tous $e, x_1, \ldots, x_m, y_1, \ldots, y_n$,*

$$\Phi_{S_n^m(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n) = \Phi_e(x_1, \ldots, x_m, y_1, \ldots, y_n).$$

Preuve. Décrivons la fonction $S_n^m$. Étant donné $e$, $x_1, \ldots, x_m$, décoder $e$ pour obtenir le programme $P_e$. Modifier calculatoirement le programme $P_e$ pour lui ajouter des instructions assignant « en dur » les valeurs $x_1, \ldots, x_m$ aux variables correspondantes, puis calculer le code du nouveau programme. Par exemple, si le programme $\Phi_e : \mathbb{N}^4 \to \mathbb{N}$ est

```
function MonProgramme(x1, x2, x3, x4) {
    // CODE
}
```

Le programme $\Phi_{S_2^2(e,5,3)}$ correspond à la chaîne

```
function MonProgramme2(x3, x4) {
    x1 = 5;
    x2 = 3;
    // CODE
}
```

■

On utilisera dans la suite le théorème SMN sans faire appel à lui explicitement, avec des phrases comme « pour tout $x$, soit $e_x$ le code du programme qui prend $y$ en entrée et fait ... [quelque chose qui dépend de $x$ et $y$] », étant entendu alors que le processus pour obtenir $e$ à partir de $x$ est calculable. On dit aussi dans ce cas que le processus est *uniforme* en $x$.

**Codage acceptable**

Le théorème d'existence d'un programme universel et le théorème SMN ne sont pas valides pour n'importe quelle fonction de codage. Rappelons ici celle que nous utilisons : chaque programme $P$ est codé par l'entier correspondant à la représentation binaire de la chaîne de caractère qui contient $P$.

Rogers [183] a prouvé que toute fonction de codage satisfaisant le théorème de programme universel et le théorème SMN était le résultat d'une permutation calculable de ce codage canonique. Il existe cependant d'autres codages des fonctions partielles calculables ne satisfaisant pas ces théorèmes. En particulier, Friedberg [64] a défini un codage calculable de toutes les fonctions partielles calculables sans répétition. Cela n'est bien entendu pas le cas pour notre codage, pour lequel une même fonction partielle a une infinité de codes différents. C'est ce que nous nous apprêtons à voir avec le lemme 5.1.

## 5. Lemme de remplissage

Le lemme de remplissage est utile de temps à autre et indique que pour tout programme informatique $e$, il existe une infinité de programmes équivalents, dont on peut de surcroît calculer le code à partir de $e$ : il suffit de rajouter des instructions qui ne servent à rien.

**Lemme 5.1 (Lemme de remplissage)**
Il existe une fonction calculable totale $h : \mathbb{N}^2 \to \mathbb{N}$ telle que pour tous $e, n \in \mathbb{N}$, on a $h(e,n) \geqslant n$, et $\Phi_{h(e,n)} = \Phi_e$. ⋆

Preuve
Étant donné le code $e$ et un entier $n$, décoder $e$ pour obtenir le programme $P_e$. Ajouter $n$ instructions inutiles à $P_e$, puis coder le nouveau programme en un entier $i$.

Par exemple, si le langage possède une instruction `skip` qui n'effectue rien, alors il suffit d'ajouter au programme une suite d'instructions `skip` comme suit :

```
function MonProgramme(x) {
    // CODE
    skip;
    skip;
    ...
}
```

■

Notons bien que $\Phi_{h(e,n)}$ et $\Phi_e$ sont égales en tant que fonctions mathématiques, mais ont des codes informatiques différents.

# 6. Théorème du point fixe de Kleene

Le théorème du point fixe de Kleene, appelé parfois *recursion theorem* en anglais, est bien plus subtil que le théorème SMN. Stephen Cole Kleene est considéré avec Kurt Gödel, Alan Turing, Emil Post et Alonzo Church, son professeur, comme un des fondateurs de la calculabilité. Il formalise avec Post la notion de *degré d'insolubilité*, que l'on appellera plus tard *degré Turing* et qui sera définie précisément dans le chapitre 4. Parmi ses travaux les plus remarquables, figurent la définition et l'étude des ensembles hyperarithmétiques et des ordinaux calculables [114], que nous verrons dans la partie IV et pour lesquels nous aurons besoin de créer des programmes informatiques *ayant accès à leur propre code.*

La notion peut sembler douteuse : comment peut-on utiliser dans la définition d'un objet $A$ l'objet $A$ lui-même ? Les auto-références mènent souvent à des paradoxes. Nous allons cependant voir que dans le cas de programmes informatiques, l'accès à son propre code est tout à fait valide, et même bien utile dans certains cas. Nous verrons un exemple remarquable d'utilisation du théorème du point fixe dans la preuve du théorème 19-1.7. Voyons plus précisément de quoi il s'agit. Le théorème stipule que pour toute fonction qui modifie des programmes, il existe un programme dont le comportement n'est pas modifié par la fonction.

Stephen Cole Kleene, 1909–1994

**Théorème 6.2**
*Pour toute fonction totale calculable $f : \mathbb{N} \to \mathbb{N}$, il existe $e \in \mathbb{N}$ tel que pour tout $n$,*

$$\Phi_{f(e)}(n) = \Phi_e(n).$$

Avant de passer à la preuve, voyons un peu en quoi cette mystérieuse affirmation permet de créer des programmes ayant accès à leur propre code. Supposons qu'un programme $M$ utilise une variable `var` ayant été initialisée à une certaine valeur au début de son exécution. On peut alors définir une fonction totale calculable $f$ qui prend un entier $n$ en paramètre, et renvoie le code du programme $M$, qui commence son exécution avec `var` initialisée à $n$. D'après le théorème du point fixe, il y a une valeur $e$ telle que les programmes de code $e$ et $f(e)$ ont le même comportement. Donc, $e$

est un code de programme équivalent à celui du programme $M$ s'exécutant avec la variable `var` initialisée à $e$ : il y a une version de $M$ qui peut accéder à son propre code. Voici, plus formellement, ce que nous venons d'énoncer.

**Corollaire 6.3**
*Pour toute fonction partielle calculable $g : \mathbb{N}^2 \to \mathbb{N}$, il existe $e \in \mathbb{N}$ tel que pour tout $n$, on a*

$$\Phi_e(n) = g(e, n)$$

Preuve. Soit $i$ tel que $\Phi_i(x, n) = g(x, n)$ pour tous $x, n$. Par le théorème SMN (voir le théorème 4.1), pour tous $x, n$, on a $\Phi_{S_2^1(i,x)}(n) = \Phi_i(x, n)$. Soit $f$ la fonction définie par $f(x) = S_2^1(i, x)$. Par le théorème du point fixe (voir le théorème 6.2), il existe $e$ tel que, pour tout $n$, $\Phi_{f(e)}(n) = \Phi_e(n)$. En particulier, $\Phi_e(n) = \Phi_{f(e)}(n) = \Phi_i(e, n) = g(e, n)$. ∎

La preuve du théorème 6.2, bien que concise, est un peu obscure ; c'est pourquoi nous fournissons au préalable un morceau de code dont l'objectif est de donner l'intuition au programmeur de la manière dont on peut écrire un programme ayant accès à son propre code. L'exemple est ici donné dans le langage Javascript.

Dans ce qui suit, les deux points de suspension doivent chacun contenir le même code, peu importe lequel. En Javascript, les « backquotes » délimitent une chaîne de caractères sur plusieurs lignes. La fonction `replace` remplacera la première occurrence de '#' par le contenu de la variable `v`.

```
function fct() {
  let v=`
function fct() {
  let v='#'
  v = v.replace('#', v)
  ... //mon code
}`
  v = v.replace('#', v)
  ... //mon code
}
```

L'exécution de ce programme se fera avec la variable `v` ayant pour contenu le programme lui-même, à ceci près que les « backquotes » se transforment alors en simple « quotes ». Il est bien entendu possible de faire en sorte que la variable `v` contienne *exactement* le programme, mais cela rendrait l'exemple beaucoup moins compréhensible. Ayant pour objectif de donner une intuition et non une preuve, nous avons conservé les choses ainsi. Pas-

sons justement à présent à la preuve formelle de notre théorème, dans le cadre des notations et principes que nous avons définis jusque-là.

Preuve du théorème 6.2. Soit $a$ le code d'une machine à un paramètre, qui sur l'entrée $n$ renvoie le code d'une machine à un paramètre $m$, qui effectue les opérations suivantes.

(1) Elle lance le calcul de $f\big(\Phi_n(n)\big)$.

(2) Si l'on a $f\big(\Phi_n(n)\big)\downarrow$, alors elle renvoie le résultat du calcul de la machine de code $f\big(\Phi_n(n)\big)$ sur l'entrée $m$ (et sinon ne s'arrête pas).

Formellement, $a$ est tel que, pour tous $n, m \in \mathbb{N}$,

$$\Phi_{\Phi_a(n)}(m) = \Phi_{f(\Phi_n(n))}(m).$$

On fait ici un léger abus de notation : si $\Phi_n(n)\uparrow$, alors $m \mapsto \Phi_{f(\Phi_n(n))}(m)$ dénote la fonction nulle part définie. Notez que $\Phi_a$ est une fonction totale : pour tout $n$, dans le calcul de $\Phi_a(n)$, on ne cherche pas à faire les étapes (1) et (2), mais seulement à calculer le code d'une machine qui les fait La démonstration qu'un tel code $a$ existe est donnée par le théorème SMN, voici comment. La fonction $(n, m) \mapsto \Phi_{f(\Phi_n(n))}(m)$ est calculable (éventuellement partielle), et il y a donc un code $b$ tel que

$$\Phi_b(n, m) = \Phi_{f(\Phi_n(n))}(m).$$

D'après le théorème SMN, il y a une fonction totale calculable $s$ telle que $\Phi_{s(b,n)}(m) = \Phi_{f(\Phi_n(n))}(m)$. Comme $s$ est totale calculable, il y a un code $a$ tel que $\Phi_a(n) = s(b, n)$.

Le point fixe sera alors $\Phi_a(a)$. En effet, on a

$$\forall m,\ \ \Phi_{\Phi_a(a)}(m) = \Phi_{f(\Phi_a(a))}(m).$$

Cela conclut la preuve. ∎

**Théorème de point fixe et paradoxe**

Comme expliqué plus haut, le théorème du point fixe de Kleene permet de concevoir des programmes *auto-référents*, c'est-à-dire des programmes qui peuvent lire leur code au cours de l'exécution, et adapter leur comportement en fonction. La capacité à faire de l'auto-référence est souvent source de paradoxes.

Le paradoxe du barbier, par exemple, raconte l'histoire d'un barbier philanthrope qui décida de raser tous les gens qui ne se rasaient pas eux-mêmes. Dût-il se raser lui-même ? Paradoxe… En mathématiques, le paradoxe de Russel suit le même schéma : soit $E$ l'ensemble de tous les ensembles qui n'appartiennent pas à eux-mêmes. Par exemple, $\mathbb{N}$ n'est pas un entier, donc $\mathbb{N} \notin \mathbb{N}$, donc $\mathbb{N} \in E$. La question problématique est alors « est-ce que $E \in E$ ? »

Pourquoi le théorème du point fixe de Kleene n'engendre-t-il pas de paradoxe ? Si l'on essaye de suivre le même schéma que les deux paradoxes précédents, on va définir une fonction $\Phi_e$ qui connaît son code $e$, et donc peut décider, pour tout entrée $n$, d'exécuter $\Phi_e(n)$ et de renvoyer une valeur différente de celle renvoyée par $\Phi_e(n)$. On aurait donc $\Phi_e(n) \neq \Phi_e(n)$. La solution vient de la partialité des fonctions : en effet, $\Phi_e$ ne sera tout simplement pas définie en $n$ et s'exécutera à l'infini.

Le lecteur désireux d'explorer les possibilités du théorème du point fixe pourra s'atteler à l'exercice suivant, dont l'objet est de démontrer le théorème de Rice, qui sera abordé plus en détail dans la section 5-6.

**Exercice 6.4. (⋆)** Soit $A \subseteq \mathbb{N}$ tel que $A \neq \mathbb{N}$, $A \neq \emptyset$ et tel que $A$ est calculable.

1. Montrer qu'il existe une fonction calculable $f$ telle que l'on a $x \in A$ ssi $f(x) \notin A$.
2. En utilisant le théorème du point fixe, en déduire qu'il existe $i, j$ tels que $\Phi_i = \Phi_j$, avec $i \in A$ et $j \notin A$.
3. En déduire qu'il n'existe pas de prédicats calculables $A$ tels que $e \in A$ ssi $e$ est le code d'un programme qui fait la multiplication par 2.
4. Généraliser la question précédente pour montrer le *théorème de Rice* : pour tout « comportement non trivial », il n'est pas possible de calculer l'ensemble des codes de programmes ayant ce comportement. Formellement : soit un prédicat $P \subseteq \mathbb{N}$ avec $P \neq \mathbb{N}$ et $P \neq \emptyset$ tel que, pour tous $e_1, e_2$ pour lesquels $\Phi_{e_1} = \Phi_{e_2}$, on a $e_1 \in P \leftrightarrow e_2 \in P$. Alors, $P$ n'est pas calculable. ◇

# 7. Ensembles calculatoirement énumérables

La calculabilité place le « calculable » comme puissance calculatoire de référence. Il s'agit de la plus faible notion calculatoire que ce paradigme permet d'identifier. Comme nous l'avons vu, un ensemble $E$ est calculable s'il existe une procédure qui, étant donné un élément, indique si cet élément appartient à $E$ ou non. La procédure doit toujours s'arrêter et donner une réponse correcte.

Il existe cependant un certain nombre de problèmes mathématiques qui peuvent s'exprimer naturellement sous forme d'ensembles dont les éléments sont énumérables par une procédure calculable, mais dans le désordre. Ces ensembles sont appelés *calculatoirement énumérables*.

Par exemple, soit $E$ l'ensemble des théorèmes mathématiques. Il n'existe pas de procédure qui, étant donné une formule mathématique, renvoie vrai ou faux selon que cette formule soit prouvable ou non. Cependant, il est possible d'énumérer les théorèmes, en listant toutes les chaînes de caractères possibles, testant s'il s'agit d'une preuve valide, et si c'est le cas, en énumérant la conclusion de la preuve (nous en discuterons plus en détail dans le chapitre 9).

Nous allons maintenant définir formellement la notion d'ensemble calculatoirement énumérable sous une forme qui peut sembler éloignée de la définition informelle que nous venons de donner. Nous verrons à travers la proposition 7.2 que ces définitions coïncident.

**Définition 7.1.** Un ensemble $A \subseteq \mathbb{N}$ est *calculatoirement énumérable* (c. e.) s'il existe un code de programme $e$ tel que $n \in A \leftrightarrow \Phi_e(n)\downarrow$, pour tout $n \in \mathbb{N}$. ♢

Notons que les ensembles calculatoirement énumérables étaient historiquement appelés *récursivement énumérables*. Il est encore fréquent de voir des articles utilisant l'ancienne terminologie, et en particulier l'abréviation r. e. au lieu de c. e.

On peut voir la machine de code $e$ comme un processus qui énumère $A$ : pour chaque entier $n$, on cherche un entier $t$ tel que $\Phi_e(n)$ s'arrête en $t$ étapes de calcul. Si un tel $t$ est trouvé, on énumère alors $n$ dans notre ensemble. Sachant qu'il existe une infinité d'entiers, il convient de fixer un ordre d'exécution pour ne réaliser qu'un nombre fini de calculs à chaque étape. L'idée est de décomposer étape par étape comme suit.

1. Tester si $\Phi(0)[0]\downarrow$.
2. Tester si $\Phi(0)[1]\downarrow$ ou $\Phi(1)[1]\downarrow$.
3. Tester si $\Phi(0)[2]\downarrow$ ou $\Phi(1)[2]\downarrow$ ou $\Phi(2)[2]\downarrow$.
4. ...

La première fois que l'on trouve une étape $t$ telle que $\Phi(n)[t]\downarrow$ pour un certain entier $n$, on énumère ce dernier. Une telle énumération peut se voir comme une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le $n$-ième élément énuméré dans $A$. Cette idée est reprise dans la preuve de la proposition suivante.

**Proposition 7.2.** Un ensemble infini $A$ est calculatoirement énumérable ssi il existe une fonction totale calculable $f : \mathbb{N} \to \mathbb{N}$ injective telle que

$$f(\mathbb{N}) = A.$$

⋆

PREUVE. Soit $A$ un ensemble calculatoirement énumérable infini, et soit $e$ tel que $\Phi_e(n)\downarrow$ ssi $n \in A$. On définit la fonction calculable $f : \mathbb{N} \to \mathbb{N}$ comme suit.

- ▷ Pour calculer $f(0)$, on cherche le plus petit $t$ tel que $\Phi_e(s)[t]\downarrow$ pour un certain $s \leqslant t$, et l'on renvoie alors le plus petit tel entier $s \leqslant t$.
- ▷ Pour calculer $f(n{+}1)$, on lance d'abord le calcul de $f(i)$ pour tout $i \leqslant n$, puis on cherche le plus petit $t$ tel que $\Phi_e(s)[t]\downarrow$ pour un certain $s \leqslant t$ tel que $s$ est différent de chaque $f(i)$ pour $i \leqslant n$, et l'on renvoie alors le plus petit tel entier $s \leqslant t$.

Le processus pour déterminer $f(n)$ est bien calculable, et comme notre ensemble calculatoirement énumérable est infini, la fonction $f$ s'arrêtera sur toutes ses valeurs. Il est alors clair que $f(\mathbb{N}) = A$.

Supposons à présent que $f(\mathbb{N}) = A$ pour une fonction totale calculable $f$. On définit la fonction $g$ qui sur $n$ cherche le plus petit $t$ tel que $f(t) = n$, et ensuite s'arrête (et sinon cherche indéfiniment sans jamais s'arrêter). On a bien $g(n)\downarrow$ ssi $\exists t\ f(t) = n$. ∎

Il devrait être clair pour le lecteur qu'un ensemble calculable est calculatoirement énumérable.

**Exercice 7.3.** Montrer que tout ensemble calculable est calculatoirement énumérable. ◇

Nous allons voir que l'inverse n'est pas forcément vrai : il y a des ensembles calculatoirement énumérables qui ne sont pas calculables. La proposition suivante donne plus précisément la connexion entre ces deux concepts.

**Proposition 7.4.** Un ensemble $A$ est calculable ssi $A$ et $\mathbb{N} \setminus A$ sont tous les deux calculatoirement énumérables. ⋆

PREUVE. Soit $A$ un ensemble calculable. D'après l'exercice 7.3, il est calculatoirement énumérable. Par ailleurs, l'ensemble $\mathbb{N} \setminus A$ est lui aussi calculable, et par conséquent calculatoirement énumérable.

Supposons à présent que l'on ait deux codes $e_1, e_2$ tels que $\Phi_{e_1}(n)\downarrow$ ssi $n \in A$ et $\Phi_{e_2}(n)\downarrow$ ssi $n \in \mathbb{N} \setminus A$. On définit la fonction calculable $f(n)$ qui cherche le plus petit $t$ tel que $\Phi_{e_1}(n)[t]\downarrow$ ou tel que $\Phi_{e_2}(n)[t]\downarrow$, puis renvoie 1 dans le premier cas et 0 dans le deuxième. Il est clair que la fonction $f$ calcule l'ensemble $A$. ∎

Si un ensemble calculatoirement énumérable n'est pas en général calculable, il peut être approximé par une suite croissante d'ensembles uniformément calculables, au sens suivant. On dit qu'une suite d'ensembles $A_0, A_1, \ldots$ est *uniformément calculable* s'il existe une fonction $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ calculable et telle que, pour tous $x, n$, on a $f(x, n) = 1$ ssi $x \in A_n$.

**Définition 7.5.** Une *approximation c. e.* d'un ensemble $A$ est une suite uniformément calculable d'ensembles $A_0, A_1, \ldots$ telle que $A_n \subseteq A_{n+1}$ pour tout $n$, et $\bigcup_n A_n = A$. ♢

**Proposition 7.6.** Un ensemble $A$ est calculatoirement énumérable si, et seulement si, il possède une approximation c. e. ⋆

Preuve. Supposons que $A$ soit c. e. Par définition, il existe un code de programme $e$ tel que $x \in A \leftrightarrow \Phi_e(x)\downarrow$, pour tout $x \in \mathbb{N}$. Soit $A_0, A_1, \ldots$ la suite uniformément calculable d'ensembles définie par $A_n = \{x : \Phi_e(x)[n]\downarrow\}$. Il est clair que $A_n \subseteq A_{n+1}$, car si la machine $\Phi_e$ s'arrête sur une entrée $x$ avant $n$ étapes, elle s'arrêtera avant $n+1$ étapes.

Supposons maintenant que l'ensemble $A$ possède une approximation c. e., à savoir $A_0, A_1, \ldots$ Soit le programme $\Phi_e$ qui, pour une entrée $x$, calcule $A_0(x)$, puis $A_1(x)$, puis $A_2(x)$, et ainsi de suite, jusqu'à ce que $A_n(x) = 1$ pour un $n$ et s'arrête à ce moment-là. Si $A_n(x) = 0$ pour tout $n$, alors $\Phi_e(x)\uparrow$, sinon $\Phi_e(x)\downarrow$. Par construction, $\{x : \Phi_e(x)\downarrow\} = \bigcup_n A_n = A$. ∎

**Notation**

Étant donné un ensemble c. e. $A$, on notera $A[0], A[1], \ldots$ une approximation c. e. de $A$ fixée. En particulier, $A[s]$ est l'*approximation de $A$ au temps $s$*. Par convention, $A[s]$ est un ensemble fini avec $\max A[s] < s$ si $A[s] \neq \emptyset$. On utilisera parfois aussi la notation $A_s$ à la place de $A[s]$.

Comme nous l'avons dit, il existe des ensembles calculatoirement énumérables qui ne sont pas calculables. L'exemple canonique est connu sous le nom de « problème de l'arrêt ».

**Définition 7.7.** On appelle le *problème de l'arrêt* ou plus simplement *l'arrêt*, que l'on note $\emptyset'$, l'ensemble :

$$\emptyset' = \{n \in \mathbb{N} \ : \ \Phi_n(n)\downarrow\}.$$ ♢

Alan Turing démontre en 1936 que l'arrêt n'est pas calculable, en utilisant un argument diagonal, comme celui introduit par Cantor pour démontrer la non-dénombrabilité des réels. Avant d'en donner une preuve mathématique, nous en donnons l'intuition via un peu de code utilisant la syntaxe Java.

Supposons par l'absurde que nous ayons à notre disposition une fonction `boolean Halt(String p, String v)` qui renvoie `true` si la méthode se trouvant dans la chaîne `p` s'arrête quand elle est exécutée avec le paramètre `v`, et renvoie `false` sinon. Comme d'habitude, si `p` contient une

chaîne de caractère ne correspondant pas à une fonction valide ou ne correspondant pas à une fonction prenant un paramètre de type `String`, alors `Halt` renvoie `false`. Considérons le programme suivant.

```
function Diagonale(x) {
   if (Halt(x, x)) {
      while (true); //boucle infinie
   } else {
      return; //fin
   }
}
```

Que renvoie l'exécution du programme `Diagonale` avec comme paramètre une chaîne de caractère `x` correspondant au programme `Diagonale` lui-même ? On voit que l'on arrive à un paradoxe :

- ▷ si `Halt(x,x)` renvoie `true`, alors `Diagonale` ne s'arrête pas sur `Diagonale` comme paramètre ;
- ▷ dans le cas inverse, `Diagonale` s'arrête sur `Diagonale` comme paramètre.

Ainsi, la méthode `Halt` ne tient pas ses promesses.

**Théorème 7.8**
*L'arrêt est un ensemble calculatoirement énumérable qui n'est pas calculable.*

Preuve. La fonction partielle $f : n \mapsto \Phi_n(n)$ est clairement calculable, et l'on a $f(n)\downarrow$ ssi $\Phi_n(n)\downarrow$. Aussi a-t-on, par définition, $\emptyset' = \{n \ : \ f(n)\downarrow\}$. Donc, $\emptyset'$ est calculatoirement énumérable.

Supposons maintenant par l'absurde que $\emptyset'$ soit un ensemble calculable. En particulier, d'après la proposition 7.4, l'ensemble $\mathbb{N} \setminus \emptyset'$ est calculatoirement énumérable, et il existe un code $e$ tel que $n \in \mathbb{N} \setminus \emptyset'$ ssi $\Phi_e(n)\downarrow$. Alors, pour tout $n$,

$$\Phi_e(n)\downarrow \ \leftrightarrow \ \ n \in \mathbb{N} \setminus \emptyset' \ \ \leftrightarrow \ \ \Phi_n(n)\uparrow .$$

En particulier, pour $n = e$,

$$\Phi_e(e)\downarrow \ \leftrightarrow \ \ e \in \mathbb{N} \setminus \emptyset' \ \ \leftrightarrow \ \ \Phi_e(e)\uparrow,$$

ce qui est une contradiction. L'arrêt n'est donc pas calculable, et en particulier le complémentaire de l'arrêt n'est pas calculatoirement énumérable. ■

Nous invitons le lecteur à considérer les exercices suivants, qui permettent de réfléchir un peu sur les ensembles calculatoirement énumérables.

**Exercice 7.9. (⋆)** Un ensemble infini est *calculatoirement énumérable dans l'ordre* s'il existe une fonction totale $f : \mathbb{N} \to \mathbb{N}$ telle que $f(\mathbb{N}) = A$ et telle que $f(n) < f(n+1)$. Montrer que si un ensemble infini $A$ est calculatoirement énumérable dans l'ordre, alors $A$ est calculable. ◇

**Exercice 7.10. (⋆)** Montrer que tout ensemble calculatoirement énumérable infini contient un sous-ensemble calculable infini. ◇

**Exercice 7.11. (⋆⋆)** Deux ensembles $A$ et $B$ sont *calculatoirement inséparables* si $A \cap B = \emptyset$ et si aucun ensemble calculable $C$ ne permet de séparer $A$ et $B$ ; autrement dit, aucun ensemble calculable $C$ n'est tel que $A \subseteq C$ et $C \cap B = \emptyset$. Montrer qu'il existe deux ensembles c. e. calculatoirement inséparables. ◇

**Exercice 7.12. (⋆⋆)** Montrer qu'étant donné $A, B \subseteq \mathbb{N}$ deux ensembles calculables, l'ensemble $D_{A,B} = \{x - y \ : \ x \geqslant y \text{ et } x \in A \text{ et } y \in B\}$ n'est pas nécessairement calculable.

Indication.– Montrer que pour tout ensemble c. e. $C \subseteq \mathbb{N}$, il existe deux ensembles calculables $A, B \subseteq \mathbb{N}$ tels que $x \in C$ ssi $2^x \in D_{A,B}$. ◇

Certaines personnes — une petite minorité, rassurez-vous — auront peut-être intégré avec une très grande facilité tout ce qui a été vu jusqu'ici, au point sans doute de s'ennuyer un peu. C'est à celles-là que s'adresse l'exercice suivant, qui devrait les occuper un petit moment…

**Exercice 7.13. (⋆⋆⋆) (*Friedberg [64]*).** Un ensemble c. e. $X$ est *maximal* si $\mathbb{N} \setminus X$ est infini et si tout ensemble c. e. $Y \supseteq X$ est tel que $Y \setminus X$ est fini ou tel que $\mathbb{N} \setminus Y$ est fini. Montrer qu'il existe un ensemble c. e. maximal.

Indication.– On note $W_e$ l'ensemble c. e. donné par $\{n \in \mathbb{N} \ : \ \Phi_e(n)\downarrow\}$. On pourra commencer par trouver un processus uniforme qui sur chaque $e \in \mathbb{N}$ associe un code $d$ tel que $\mathbb{N} \setminus W_d$ est infini, et tel que si $W_d \subseteq W_e$ ; alors, soit $W_e$ est fini, soit $\mathbb{N} \setminus W_e$ est fini. ◇

Chapitre 4

# Degrés Turing

Un ensemble non calculable peut être vu comme un problème insoluble : il n'existe pas d'algorithme permettant de décider si un entier appartient ou non à cet ensemble. Un des objectifs de la calculabilité est d'étudier et de comprendre l'univers des problèmes insolubles, via différentes comparaisons et classifications. On introduit pour cela différents outils. Le plus important d'entre eux est l'objet de ce chapitre et trouve sa genèse dans « Systems of logic based on ordinals » [226], le fameux article d'Alan Turing présentant ses travaux de thèse, et sera formalisé et étudié plus tard par Post [181] puis Post et Kleene [118] : étant donné un ensemble non calculable $A$, on imagine pouvoir l'utiliser comme « oracle » afin d'augmenter la puissance de calcul de nos machines. On dira alors que deux ensembles sont dans le même *degré d'insolubilité* ou dans le même *degré Turing*, si chacun peut se calculer avec un algorithme utilisant l'autre comme « oracle ».

## 1. Les chaînes finies

Avant d'entrer dans le vif du sujet, nous devons introduire un peu de vocabulaire et de notation sur les chaînes binaires. Formellement, une chaîne binaire est une fonction partielle de $\mathbb{N}$ vers $\{0, 1\}$ dont le domaine de définition est un segment initial de $\mathbb{N}$. De manière plus informelle, il s'agit d'une suite finie de 0 et de 1.

**Définition 1.1.** On note $2^{<\mathbb{N}}$ l'ensemble des suites finies de 0 et de 1. Les variables $\sigma, \tau, \rho$ seront normalement utilisées pour dénoter des éléments de $2^{<\mathbb{N}}$, que l'on appellera généralement des *chaînes*.

Ces suites seront manipulées via les symboles suivants :

- ▷ $\epsilon$ : la chaîne vide, de taille 0
- ▷ $i^n$ pour $i \in \{0,1\}$ : une suite de $n$ répétitions du bit $i$
- ▷ $\sigma\tau$ ou $\sigma \frown \tau$ : la concaténation de $\sigma$ et $\tau$
- ▷ $\sigma \preceq \tau$ : la chaîne $\sigma$ est un préfixe de $\tau$, c'est-à-dire $\exists\rho$ tel que $\sigma\rho = \tau$
- ▷ $\sigma \prec \tau$ : la chaîne $\sigma$ est un préfixe strict de $\tau$, c'est-à-dire $\exists\rho \neq \epsilon$ tel que $\sigma\rho = \tau$
- ▷ $|\sigma|$ : la taille de $\sigma$
- ▷ $\sigma(n)$ pour $n < |\sigma|$ : la valeur du n-ième bit de $\sigma$, en commençant à 0
- ▷ On dira que deux chaînes $\sigma, \tau$ sont *incompatibles* si l'on n'a ni $\sigma \preceq \tau$ ni $\tau \preceq \sigma$ (on écrira aussi $\sigma \npreceq \tau$ et $\tau \npreceq \sigma$). Si à l'inverse $\sigma \preceq \tau$ ou $\tau \preceq \sigma$, les deux chaînes $\sigma$ et $\tau$ sont compatibles. ♢

On identifiera parfois un bit $i \in \{0,1\}$ avec la chaîne de longueur 1 dont l'unique bit est $i$. Ainsi, on notera $\sigma i$ ou $\sigma \frown i$ la concaténation d'une chaîne $\sigma$ et d'un bit $i$. Suivant les définitions précédentes, $|\epsilon| = 0$, et pour toute chaîne $\sigma$ non vide, le premier et dernier bit sont respectivement $\sigma(0)$ et $\sigma(|\sigma| - 1)$. Les chaînes et les suites infinies peuvent se combiner.

**Définition 1.2.** On adoptera les notations suivantes pour $\sigma \in 2^{<\mathbb{N}}$ et $X \in 2^{\mathbb{N}}$ :

- ▷ $\sigma X$ : la concaténation de $\sigma$ et $X$
- ▷ $\sigma \prec X$ : la chaîne $\sigma$ est un préfixe de $X$, c'est-à-dire $\exists Y \in 2^{\mathbb{N}}\ \sigma Y = X$
- ▷ $X\restriction_n$ pour $n \geqslant 0$ : le préfixe de $X$ de taille $n$. ♢

## 2. Calcul avec oracle

Intuitivement, un calcul avec oracle, par exemple $A \subseteq \mathbb{N}$, est simple à comprendre : il s'agit d'un calcul qui peut à tout moment utiliser une instruction qui « pose une question » à l'oracle, de la forme : « est-ce que $n$ appartient à $A$ ? » Cette instruction peut s'apparenter à un appel de fonction, qui renvoie toujours la bonne réponse.

Si l'oracle utilisé n'est pas calculable, on peut donc écrire des programmes informatiques qui calculent, à l'aide de cet oracle, des objets normalement non calculables, à commencer par l'oracle lui-même. Notre objectif est à présent d'étudier les ensembles d'entiers du point de vue de la puissance de calcul qu'ils fournissent quand ils sont utilisés comme oracles.

**Définition 2.1.** Une *fonctionnelle* Turing ou simplement une *fonctionnelle* est une fonction calculable par un algorithme ayant la possibilité d'utiliser, en plus de son jeu d'instruction habituel, une instruction permettant de savoir si un entier $n$ appartient à l'oracle. ♢

Une fonctionnelle prend donc un ou plusieurs paramètres entiers, comme pour les fonctions calculables, et un ou plusieurs paramètres d'oracles, ici des ensembles d'entiers (on parle aussi parfois de paramètres « réels » quand on les voit comme le développement binaire de nombres réels). On appellera aussi *paramètres du premier ordre* les paramètres entiers et *paramètres du second ordre* les paramètres qui sont des ensembles d'entiers.

**Notation**

On note $\Phi_e(A, n)$ ou $\Phi_e^A(n)$ le résultat du calcul de la fonctionnelle $\Phi_e$ avec l'oracle $A$ et sur l'entrée $n$. On écrira de la même manière $\Phi_e(A, n)\downarrow$, $\Phi_e(A, n)\uparrow$, $\Phi_e(A, n)[t]\downarrow$, $\Phi_e(A, n)[t]\uparrow$ pour signifier que la fonctionnelle $\Phi_e$ respectivement s'arrête, ne s'arrête pas, s'arrête en temps inférieur à $t$, ne s'arrête pas en temps inférieur à $t$, avec l'oracle $A$ et sur l'entrée $n$.

Afin d'avoir une vision uniforme, on suppose à présent que l'on ne travaille qu'avec des fonctionnelles. Il est facile de voir que les fonctions calculables sont exactement celles qui le sont par des fonctionnelles utilisant l'ensemble vide comme oracle (ou n'importe quel autre ensemble calculable).

**Définition 2.2.** Un ensemble $A$ est dit *$X$-calculable* ou calculable relativement à $X$ s'il est calculable par une fonctionnelle Turing utilisant $X$ comme oracle. Un ensemble $A$ est dit *calculatoirement énumérable relativement à $X$* si c'est le domaine de définition d'une fonctionnelle Turing utilisant $X$ comme oracle. ♢

La notion d'oracle se généralise aux fonctions. Avec un oracle $f : \mathbb{N} \to \mathbb{N}$, un calcul consiste à ajouter la fonction $f$ aux primitives du langage de programmation. Ainsi, le programme pourra à tout moment interroger $f$ sur des entrées pour en connaître les résultats. On peut par conséquent parler d'ensemble *$f$-calculable* s'il est calculable par une fonctionnelle Turing utilisant $f$ comme oracle. De manière équivalente, un ensemble est $f$-calculable s'il est $G_f$-calculable, où $G_f \in 2^{\mathbb{N}}$ est un encodage du graphe de la fonction $f$ par un élément de $2^{\mathbb{N}}$, par exemple avec $\langle n, m\rangle \in G_f$ ssi $f(n) = m$.

**Notation**

On écrira aussi $X$-c. e. pour signifier calculatoirement énumérable relativement à $X$.

**Exemple 2.3.** Supposons par exemple que l'on dispose d'un oracle $X$ inclus dans $\mathbb{N}$ tel que la fonction $f$ qui à $n$ associe le $n$-ième élément de $X$ croisse suffisamment vite pour borner le temps d'arrêt des programmes informatiques. Formellement : $\Phi_e(e)\downarrow$ implique $\Phi_e(e)[f(e)]\downarrow$ pour tout $e \in \mathbb{N}$. Alors, il est aisé de créer une fonctionnelle Turing permettant de calculer $\emptyset'$ à partir de $X$ : pour savoir si $e \in \emptyset'$, il suffit de parcourir $X$ jusqu'à trouver son $e$-ième élément $f(e)$, puis de calculer $\Phi_e(e)$ durant $f(e)$ étapes de calcul. Si $\Phi_e(e)[f(e)]\downarrow$, alors $e \in \emptyset'$. Sinon, $e \notin \emptyset'$.

## 3. Relativisation des preuves

La plupart des arguments de calculabilité s'appliquent aux machines à oracle en remplaçant « machine » par « machine avec un oracle $X$ ». Cette opération purement syntaxique que l'on appelle *relativisation* (à un oracle) donne donc gratuitement un schéma de résultats similaires, paramétrés par un oracle $X$. Prenons l'exemple de l'indécidabilité du problème de l'arrêt, en remplaçant la notion de calcul par celle de calcul avec oracle $X$.

**Théorème 3.1**
*Pour tout oracle* $\mathbf{X}$*, l'ensemble* $Y = \{n : \Phi_n^{\mathbf{X}}(n)\downarrow\}$ *n'est pas* $\mathbf{X}$*-calculable.*

Preuve. La fonction partielle $f : n \mapsto \Phi_n^{\mathbf{X}}(n)$ est clairement $\mathbf{X}$-calculable, et l'on a $f(n)\downarrow$ ssi $\Phi_n^{\mathbf{X}}(n)\downarrow$. Aussi a-t-on, par définition, $Y = \{n \in \mathbb{N} : f(n)\downarrow\}$. Donc, $Y$ est $\mathbf{X}$-calculatoirement énumérable.

Supposons maintenant par l'absurde que l'ensemble $Y$ est $\mathbf{X}$-calculable.En particulier, d'après la proposition 3-7.4 **relativisée à X**, l'ensemble $\mathbb{N} \setminus Y$ est $\mathbf{X}$-calculatoirement énumérable, et il existe un code $e$ tel que $n \in \mathbb{N} \setminus Y$ ssi $\Phi_e^{\mathbf{X}}(n)\downarrow$. On a alors pour tout $n$

$$\Phi_e^{\mathbf{X}}(n)\downarrow \leftrightarrow \ n \in \mathbb{N} \setminus Y \ \leftrightarrow \ \Phi_n^{\mathbf{X}}(n)\uparrow .$$

En particulier, pour $n = e$, on a

$$\Phi_e^{\mathbf{X}}(e)\downarrow \leftrightarrow \ e \in \mathbb{N} \setminus Y \ \leftrightarrow \ \Phi_e^{\mathbf{X}}(e)\uparrow,$$

ce qui est une contradiction. Donc, $Y$ n'est pas $\mathbf{X}$-calculable, et en particulier le complémentaire de $Y$ n'est pas $\mathbf{X}$-calculatoirement énumérable. ∎

Il convient cependant d'être précautionneux lors de la relativisation d'un argument. En effet, certaines définitions masquent des appels à des machines, et leur définition doit alors être également relativisée. Par exemple, si l'on

définit le *problème de l'arrêt* comme l'ensemble $\{n : \Phi_n(n)\downarrow\}$, la relativisation de l'énoncé « Le problème de l'arrêt n'est pas calculable » n'est pas « Pour tout $X$, le problème de l'arrêt n'est pas $X$-calculable », mais « Pour tout $X$, le problème de l'arrêt relativisé à $X$ n'est pas $X$-calculable », où le « problème de l'arrêt relativisé à $X$ » est en fait défini comme l'ensemble $\{n : \Phi_n^X(n)\downarrow\}$.

De manière générale, la relativisation d'un théorème s'obtient en ajoutant un oracle $X$ à chaque machine, et en remplaçant *calculable* par *$X$-calculable*. C'est par exemple le cas pour une relativisation naïve du théorème SMN.

**Théorème 3.2 (Théorème SMN relativisé - version 1)**
***Pour tout oracle* X**, *et pour tous entiers $n$ et $m$ non nuls, il existe une fonction* **X-calculable** *totale $S_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ telle que pour tout $e$,*

$$\Phi^{\mathbf{X}}_{S_n^m(e,x_1,\ldots,x_m)}(y_1,\ldots,y_n) = \Phi^{\mathbf{X}}_e(x_1,\ldots,x_m,y_1,\ldots,y_n).$$

Une analyse de la preuve du théorème SMN révèle cependant que la fonction $S_n^m$ ne dépend pas de l'oracle $X$, car elle effectue des manipulations purement syntaxiques sur la machine. Il est donc possible de formuler une version relativisée plus forte du théorème SMN, où la fonction $S_n^m$ est calculable, bien que la version précédente soit également valide.

**Théorème 3.3 (Théorème SMN relativisé - version 2)**
***Pour tout oracle* X**, *pour tous $n, m \in \mathbb{N}^*$ il existe une fonction* ***calculable*** *totale $S_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ telle que pour tout $e$,*

$$\Phi^{\mathbf{X}}_{S_n^m(e,x_1,\ldots,x_m)}(y_1,\ldots,y_n) = \Phi^{\mathbf{X}}_e(x_1,\ldots,x_m,y_1,\ldots,y_n).$$

**Digression**

La relativisation des arguments est un phénomène empirique qui d'une certaine manière reflète notre compréhension partielle de la notion de calcul. Toutefois, et notamment en théorie de la complexité, les preuves ne se relativisent pas nécessairement. L'exemple emblématique est la question de la séparation des classes de complexité P et NP. Chacune de ces classes peut se relativiser à un oracle. Baker, Gill, et Solovay [9] ont montré qu'il existait des oracles $X$ pour lesquels $\mathrm{P}^{\mathrm{X}} = \mathrm{NP}^{\mathrm{X}}$ et d'autres pour lesquels $\mathrm{P}^{\mathrm{X}} \neq \mathrm{NP}^{\mathrm{X}}$. Il est alors nécessaire pour résoudre la question P vs NP d'utiliser un argument qui ne se relativise pas. Il s'agit en calculabilité de quelque chose de tout à fait inhabituel, mais pas nécessairement en complexité où nous avons d'autres exemples de problèmes résolus qui

ne se relativisent pas. On sait par exemple que les classes IP et PSPACE coïncident [196], tout en étant capable de produire des oracles $X$ pour lesquels $\mathrm{IP}^X \neq \mathrm{PSPACE}^X$ [60].

## 4. Propriété de l'usage

Étant donné un calcul $\Phi_e(A,n)$ sur un oracle $A$, il est clair que si l'on suppose que $\Phi_e(A,n)\downarrow$, alors la fonctionnelle $\Phi_e$ n'a utilisé qu'une partie finie de l'oracle $A$ pour renvoyer le résultat : cela découle simplement du fait qu'un calcul s'effectue toujours en un nombre fini d'étapes. Cela nous amène à définir la notion de calcul avec des oracles finis.

**Notation**

Étant donné une suite finie $\sigma \in 2^{<\mathbb{N}}$, on note $\Phi_e(\sigma,n)\downarrow$ ou $\Phi_e^\sigma(n)\downarrow$ pour signifier que le calcul s'arrête sur l'entrée $n$ et avec $\sigma$ comme morceau d'oracle, la fonctionnelle n'ayant alors besoin pour son calcul que de poser à l'oracle des questions auxquelles $\sigma$ peut répondre, c'est-à-dire des questions d'appartenance de $i$ à l'oracle pour des entiers $i$ strictement inférieurs à $|\sigma|$.

Dans la notation précédente, si le calcul a besoin d'accéder à des valeurs de l'oracle qui dépassent la taille de $\sigma$, alors on note $\Phi_e(\sigma,n)\uparrow$ ou $\Phi_e^\sigma(n)\uparrow$. La partie finie de l'oracle $A$ interrogée jusqu'à ce que le calcul $\Phi_e(A,n)$ termine s'appelle l'*usage* du calcul.

**Proposition 4.1 (Propriété de l'usage).** Soient $\Phi_e$ une fonctionnelle Turing, $X$ un oracle et $n \in \mathbb{N}$ une entrée. Alors, $\Phi_e(X,n)\downarrow$ si, et seulement si, il existe un préfixe fini $\sigma \prec X$ tel que $\Phi_e(\sigma,n)\downarrow$. ⋆

Nous verrons dans la section 8-2 que la propriété de l'usage correspond au concept de continuité sur l'espace de Cantor. Malgré sa simplicité conceptuelle, la propriété de l'usage joue un rôle primordial en calculabilité. Nous en ferons par exemple une utilisation forte dans la section 8 sur la méthode des extensions finies.

**Définition 4.2.** Étant donné une fonctionnelle $\Phi$ et un oracle $X$, on note $\mathrm{use}_\Phi^X : \mathbb{N} \to \mathbb{N}$ la fonction partielle $X$-calculable qui sur $n$ renvoie la taille minimale du préfixe de $X$ nécessaire à l'arrêt du calcul de $\Phi(X,n)$.♢

**Remarque**

D'après la propriété de l'usage, toute fonctionnelle Turing $\Gamma$ peut être représentée par l'ensemble c. e. $W$ des triplets $(\sigma,x,y)$ tels que

si $(\sigma, x, y) \in W$, alors $\Gamma^\sigma(x) \downarrow = y$. Une telle énumération satisfait la propriété de cohérence suivante : pour tous $(\sigma, x, y) \in W$ et $(\tau, x, y') \in W$ tels que $\sigma \prec \tau$, on a $y = y'$.

**Exercice 4.3.** Montrer que si $Y$ est $X$-calculable et $Z$ est $Y$-calculable, alors $Z$ est $X$-calculable. ◇

## 5. Degrés Turing

Les machines à oracle induisent une notion de calculabilité relative. Informellement, un ensemble $Y$ est $X$-calculable si $X$ est au moins aussi puissant que $Y$, au sens où tout ce qui est calculable par $Y$ l'est également par $X$. Cela donne lieu à la *réduction Turing*.

**Définition 5.1 (Réduction Turing).** Pour tous ensembles $X, Y \subseteq \mathbb{N}$, on écrit $X \leqslant_T Y$ — et l'on dit que $X$ est *Turing réductible* à $Y$ — si $X$ est $Y$-calculable. On écrit $X <_T Y$ si $X \leqslant_T Y$ mais $Y \not\leqslant_T X$. ♢

La réduction Turing forme un *pré-ordre* sur les ensembles d'entiers, c'est-à-dire que cette relation est réflexive et transitive. En effet, si $Y$ est $X$-calculable et $Z$ est $Y$-calculable, $Z$ est $X$-calculable. Ce n'est cependant pas un ordre partiel sur les ensembles, car la réduction Turing n'est pas antisymétrique. Par exemple, les ensembles des nombres pairs et des nombres impairs sont trivialement mutuellement calculables puisqu'ils sont tous deux calculables, mais ils ne sont pas égaux en tant qu'ensembles d'entiers.

Il est cependant possible de transformer ce pré-ordre en un ordre partiel en identifiant tous les ensembles mutuellement calculables. Cela donne la notion de *degré Turing* qui représente une puissance calculatoire plus robuste que la notion d'ensemble pour la réduction Turing.

**Définition 5.2.** On écrit $X \equiv_T Y$ si $X \leqslant_T Y$ et $Y \leqslant_T X$. On dira alors que $X$ et $Y$ sont *Turing-équivalents*. On appelle *degrés Turing* les classes d'équivalences de la relation $\equiv_T$. Le degré Turing d'un ensemble $X$ est l'ensemble $\deg_T(X) = \{Y : Y \equiv_T X\}$. ♢

Par construction, si $X \leqslant_T Y$, alors tout élément dans le degré Turing de $Y$ calcule n'importe quel élément dans le degré Turing de $X$. La réduction de Turing induit donc un ordre partiel sur les degrés Turing, que l'on notera tout simplement $\leqslant$.

**Notation**

On notera $(\boldsymbol{\mathcal{D}}, \leqslant)$ l'ensemble des degrés Turing $\boldsymbol{\mathcal{D}}$ partiellement ordonné par $\leqslant$. On utilisera en général les lettres $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \ldots$ pour désigner ses éléments. On écrira parfois $X \leqslant_T \mathbf{d}$ pour $\deg_T(X) \leqslant \mathbf{d}$.

**Exercice 5.3.** Montrer que si $X \equiv_T Y$ et $A \equiv_T B$, on a alors $X \leqslant_T A$ si, et seulement si, $Y \leqslant_T B$. ◇

Deux ensembles sont donc Turing-équivalents s'ils ont la même puissance calculatoire, et la relation $\leqslant$ permet de comparer non plus des ensembles, mais des degrés de puissance calculatoire. En particulier, les degrés Turing sont stables par variations finies, au sens où si $X \in \mathbf{d}$ et $Y =^* X$, alors $Y \in \mathbf{d}$. Ici, $Y =^* X$ signifie que $X$ et $Y$ ne diffèrent que sur un nombre fini de bits.

**Exercice 5.4.** Montrer que les degrés Turing sont stables par variation finie. ◇

Une grande partie de la calculabilité classique consiste à comprendre la structure $(\boldsymbol{\mathcal{D}}, \leqslant)$. L'ordre $\leqslant$ est-il total sur $\boldsymbol{\mathcal{D}}$ ? Est-il bien fondé ? Si c'est un ordre partiel, quelle est la taille maximale d'un ensemble ne contenant que des éléments deux à deux incomparables ? Nous verrons que la structure des degrés Turing est d'une grande richesse, mais aussi d'une grande complexité. Commençons par quelques observations immédiates.

Tout d'abord, les degrés Turing possèdent un élément minimal, à savoir le degré des ensembles calculables. On le notera $\mathbf{0}$. Nous avons ensuite la proposition suivante.

**Proposition 5.5.** Tout degré Turing est dénombrable. ⋆

Preuve. Soit $\mathbf{d}$ un degré Turing et soit $X \in \mathbf{d}$. En particulier,

$$\mathbf{d} = \deg_T(X) = \{Y : X \equiv_T Y\} \subseteq \big\{\{n \ : \ \Phi_e^X(n)\!\downarrow = 1\} : e \in \mathbb{N}\big\},$$

donc $\mathbf{d}$ est fini ou dénombrable. Par ailleurs, $\mathbf{d}$ est stable par variations finies, donc est infini. Ainsi, $\mathbf{d}$ est dénombrable. ∎

La collection des sous-ensembles de $\mathbb{N}$ étant indénombrable, et chacun appartenant à un degré Turing, il s'ensuit de la proposition précédente que les degrés Turing sont en quantité indénombrable. Supposons en effet le contraire. Alors, d'après l'exercice 2-3.11, la réunion de tous les degrés Turing serait un ensemble dénombrable, en tant que réunion dénombrable d'ensembles dénombrables. Comme cette réunion est égale à $2^{\mathbb{N}}$, on a là une contradiction. Remarquons que l'exercice 2-3.11 utilise l'axiome du choix (dont nous parlerons en détail dans la section 9-4). Celui-ci n'est toutefois pas nécessaire : nous verrons plusieurs constructions effectives

de fonctions $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ (voir l'exercice 8-5.4 ou l'exercice 8-5.3) telles que $f(X)$ et $f(Y)$ sont dans des degrés Turing différents pour tous $X \neq Y$, ce qui fait de $f$ une injection de $2^{\mathbb{N}}$ dans les degrés Turing et montre en particulier $|2^{\mathbb{N}}| \leqslant |\boldsymbol{\mathcal{D}}|$ [1].

**Définition 5.6.** La *jointure effective* de deux ensembles $A$ et $B$ est l'ensemble $A \oplus B = \{2n : n \in A\} \cup \{2n+1 : n \in B\}$ (notons que la réunion entre les deux ensembles est disjointe). ♢

La jointure effective de deux ensembles est une façon d'encoder l'information de chaque ensemble de manière à pouvoir la décoder calculatoirement. Il existe bien entendu de nombreuses manières d'encoder deux ensembles en un seul, la jointure effective étant la plus directe et efficace, au sens suivant.

**Proposition 5.7.** Soient $A$ et $B$ deux ensembles. Alors, $\deg_T(A \oplus B)$ est la borne supérieure des degrés $\deg_T(A)$ et $\deg_T(B)$, c'est-à-dire que tout degré au-dessus de $\deg_T(A)$ et de $\deg_T(B)$ est aussi au-dessus de $\deg_T(A \oplus B)$. Ainsi, toute paire de degrés Turing $\mathbf{c}$ et $\mathbf{d}$ admet une borne supérieure que l'on notera $\mathbf{c} \cup \mathbf{d}$. ⋆

Preuve. Il est clair que la jointure $A \oplus B$ permet de calculer $A$ et $B$. Ainsi, $\deg_T(A \oplus B)$ est un majorant de $\deg_T(A)$ et $\deg_T(B)$. Soit $\deg_T(C)$ un majorant de $\deg_T(A)$ et $\deg_T(B)$. En particulier, $C \geqslant_T A$ et $C \geqslant_T B$. Soient $i$ et $j$ des codes tels que $\Phi_i(C, n) = A(n)$ et $\Phi_j(C, n) = B(n)$ pour tout $n$. La fonction $f : \mathbb{N} \to \{0, 1\}$ définie par

$$f(n) = \begin{cases} \Phi_i(C, n/2) & \text{si n est pair} \\ \Phi_j(C, (n-1)/2) & \text{sinon} \end{cases}$$

est $C$-calculable, et l'on a $f(n) = 1$ ssi $n \in A \oplus B$ pour tout entier $n$. Donc, $C \geqslant_T A \oplus B$. ∎

**Notation**

Nous avons fait usage dans la preuve de la proposition précédente du prédicat $\ll \Phi_i(C, n) \downarrow = A(n) \gg$ pour tout $n$. Il nous arrivera quelques fois d'utiliser la notation $\Phi_i(C) = A$, plus courte.

Attention, la borne supérieure $\mathbf{c} \cup \mathbf{d}$ de $\mathbf{c}$ et $\mathbf{d}$ n'a bien entendu rien à voir avec la réunion ensembliste des degrés $\mathbf{c}$ et $\mathbf{d}$. De manière générale, les lettres en caractère gras $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \ldots$ seront utilisées pour parler des degrés en tant qu'objets abstraits au sein d'un ordre partiel, le détail des ensembles d'entiers constituant chaque degré n'étant alors pas pertinent.

1. L'inégalité $|\boldsymbol{\mathcal{D}}| \leqslant |2^{\mathbb{N}}|$ semble évidente, mais elle utilise l'axiome du choix, afin de sélectionner uniformément un élément dans chaque degré Turing. Nous en reparlerons brièvement dans la section 12-2.3.

Nous jonglerons désormais entre les ensembles d'entiers et les degrés Turing. Chaque degré Turing pouvant être représenté par un ensemble d'entiers (l'un de ses membres), une opération sur les degrés Turing se fera normalement via une opération sur l'un de ses représentants, de manière à ce que le résultat attendu soit indépendant du choix d'un tel représentant. La jointure effective constitue un premier exemple illustrant notre propos.

**Exercice 5.8.** Montrer que si l'on suppose que $X \leqslant_T Y$ et $A \leqslant_T B$, alors

$$X \oplus A \leqslant_T Y \oplus B.$$ ◇

L'exercice précédent montre que la jointure effective induit une opération sur les degrés Turing. Nous allons étudier dans la section suivante une nouvelle opération sur les degrés jouant un rôle essentiel en calculabilité : le *saut Turing*.

# 6. Saut Turing

Le saut Turing est une opération fondamentale en calculabilité, et se définit comme la relativisation du problème de l'arrêt.

**Définition 6.1.** Étant donné un ensemble $X$, on définit

$$X' = \{n \ : \ \Phi_n^X(n)\downarrow\}.$$

Le *saut Turing* est l'opérateur $X \mapsto X'$. ♢

On peut par exemple à présent définir l'arrêt relativement à l'arrêt : l'ensemble des codes de programmes informatiques qui s'arrêtent sur leur propre entrée, mais en utilisant l'arrêt en tant qu'oracle. On le note $\emptyset''$. Il n'est pas très difficile de montrer que le saut Turing induit une opération sur degrés Turing, et nous en laissons la preuve en exercice.

**Exercice 6.2.** **(⋆)** Montrer que si $X \leqslant_T Y$, alors $X' \leqslant_T Y'$. ◇

Nous verrons un renforcement du résultat de l'exercice précédent avec l'exercice 5-5.7. On notera donc $\mathbf{d}'$ le saut Turing d'un degré Turing $\mathbf{d}$. En particulier, $\mathbf{0}'$ est le degré Turing du problème de l'arrêt.

**Proposition 6.3.** On a $X <_T X'$ pour tout $X \in 2^{\mathbb{N}}$. Ainsi a-t-on $\mathbf{d} < \mathbf{d}'$ pour tout degré Turing $\mathbf{d}$. ⋆

Preuve. Montrons d'abord que $X'$ calcule $X$. Soit $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ la fonction partielle $X$-calculable définie par

$$f(e, n) = \begin{cases} 1 & \text{si } e \in X \\ \uparrow & \text{sinon.} \end{cases}$$

Par le théorème SMN relativisé à $X$ (voir le théorème 3.3), il existe une fonction totale calculable $g : \mathbb{N} \to \mathbb{N}$ telle que pour tous entiers $e$ et $n$, on a $\Phi^X_{g(e)}(n) = f(e, n)$. Ainsi, pour tout $e$ :

- ▷ si $e \in X$, alors $\Phi^X_{g(e)}$ est la fonction constante 1, et donc $g(e) \in X'$ ;
- ▷ si $e \notin X$, alors $\Phi^X_{g(e)}$ est la fonction nulle part définie, et $g(e) \notin X'$.

En particulier, $X'$ peut calculer $X$ : pour savoir si $n \in X$, il suffit de regarder si $g(n) \in X'$.

La preuve que $X \not\geqslant_T X'$ est une relativisation du fait que $\emptyset'$ n'est pas calculable, ce qui a été démontré préalablement avec le théorème 3.1. ∎

Il existe donc une hiérarchie strictement croissante de degrés Turing :

$$\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \dots$$

Dans la preuve précédente, notons que la fonctionnelle Turing utilisée pour calculer $X$ à partir de $X'$ est *la même* pour tout oracle $X$. C'est quelque chose qui sera utilisé de temps à autre, par exemple dans le chapitre 26. Nous donnons l'occasion, ci-après, au lecteur non convaincu d'y réfléchir.

**Exercice 6.4.** Montrer qu'il existe une fonctionnelle $\Phi_e$ telle que :

- ▷ $\Phi_e(X', n) = X(n)$ pour tout $X \in 2^{\mathbb{N}}$ et pour tout $n \in \mathbb{N}$.
- ▷ $\Phi_e(Y, n)\!\downarrow$ pour tout $Y \in 2^{\mathbb{N}}$ et pour tout $n \in \mathbb{N}$. ◇

Notons enfin que le saut Turing n'est pas un opérateur injectif, comme nous le verrons par la suite à travers les ensembles low et high. Nous utiliserons de temps à autre la notion suivante de Turing-complétude.

**Définition 6.5.** Un ensemble $A$ est dit *Turing-complet* ou (simplement) *complet* si $A \geqslant_T \emptyset'$. Un degré Turing *complet* est un degré $\mathbf{d} \geqslant \mathbf{0}'$. Un ensemble ou degré qui n'est pas complet est *incomplet*. ♢

# 7. Calculabilité à la limite

Nous allons maintenant étudier certaines propriétés des ensembles calculables par le problème de l'arrêt. Ces ensembles admettent notamment une caractérisation très naturelle en termes d'approximations.

**Définition 7.1.** Une fonction $f : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ est *stable* si pour tout $x \in \mathbb{N}$, $\lim_y f(x, y)$ existe. Un ensemble $A$ est *calculable à la limite* s'il existe une fonction stable calculable $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ telle que pour tout $x$

$$\lim_y f(x, y) = 1 \quad \Leftrightarrow \quad x \in A.$$

♢

**Lemme 7.2 (Lemme de limite de Shoenfield).** Un ensemble $A \subseteq \mathbb{N}$ est $\emptyset'$-calculable si, et seulement si, il est calculable à la limite. ⋆

Preuve. On pourra se reporter aux figures 7.4 et 7.3 pour s'aider dans la compréhension de la preuve.

$\Rightarrow$. Supposons que $A \leqslant_T \emptyset'$ via une fonctionnelle $\Phi_e$. Soit $\emptyset'_0 \subseteq \emptyset'_1 \subseteq \ldots$ une approximation c. e. de $\emptyset'$, et soit $f : \mathbb{N}\times\mathbb{N} \to \{0,1\}$ la fonction qui pour une entrée $(x,s)$ regarde si $\Phi_e(\emptyset'_s, x)[s]\downarrow$. Si tel est le cas, $f(x,s) = \Phi_e(\emptyset'_s, x)[s]$. Sinon, $f(x,s)$ se voit attribuer une valeur arbitraire.

Montrons que $f$ est stable et que sa limite est $A$. Soit $x \in \mathbb{N}$ ; par la propriété de l'usage, comme $\Phi_e(\emptyset', x)\downarrow$, ce calcul est effectué en utilisant les $n$ premiers bits de l'oracle pour un certain entier $n$. Soit alors $s$ tel que $\emptyset'_s \upharpoonright_n = \emptyset' \upharpoonright_n$ et tel que $\Phi_e(\emptyset', x)\downarrow$ s'arrête après $s$ étapes de calcul (il suffit de prendre $s$ suffisamment grand). Alors, pour tout $t \geqslant s$,

$$\Phi_e(\emptyset'_t, x)[t]\downarrow = \Phi_e(\emptyset', x),$$

auquel cas $\lim_t f(x,t) = \Phi_e(\emptyset', x) = A(x)$.

$\Leftarrow$. Supposons maintenant que $A$ est calculable à la limite, par une fonction stable calculable $f : \mathbb{N}\times\mathbb{N} \to \{0,1\}$. Alors,

$$A = \{x : \exists y \forall z \geqslant y \ f(x,z) = 1\} \quad \text{et} \quad \overline{A} = \{x : \exists y \forall z \geqslant y \ f(x,z) = 0\}.$$

Nous allons définir une procédure $\emptyset'$-calculable pour déterminer si $x \in A$ ou $x \in \overline{A}$. Soient $u, v : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ deux fonctions totales calculables telles que pour tous $x, y, n$,

$$\Phi_{u(x,y)}(n) = \begin{cases} 1 & \text{si } \exists z \geqslant y \ f(x,z) \neq 1 \\ \uparrow & \text{sinon,} \end{cases}$$

$$\Phi_{v(x,y)}(n) = \begin{cases} 1 & \text{si } \exists z \geqslant y \ f(x,z) \neq 0 \\ \uparrow & \text{sinon.} \end{cases}$$

En particulier,

$$u(x,y)\notin\emptyset' \ \text{ ssi } \ \forall z \geqslant y \ f(x,z) = 1 \quad \text{et} \quad v(x,y)\notin\emptyset' \ \text{ ssi } \ \forall z \geqslant y \ f(x,z) = 0.$$

Ainsi,
$$A = \{x : \exists y \ u(x,y) \notin \emptyset'\} \quad \text{et} \quad \overline{A} = \{x : \exists y \ v(x,y) \notin \emptyset'\}.$$

Étant donné un entier $x$, pour savoir si $x \in A$ ou $x \in \overline{A}$, il suffit de chercher le plus petit $y$ tel que $u(x,y) \notin \emptyset'$ ou $v(x,y) \notin \emptyset'$. On finira nécessairement par trouver un tel entier $y$. Si $u(x,y) \notin \emptyset'$, alors $x \in A$. Si $v(x,y) \notin \emptyset'$, alors $x \notin A$. La procédure est $\emptyset'$-calculable, et l'on a ainsi $A \leqslant_T \emptyset'$. ∎

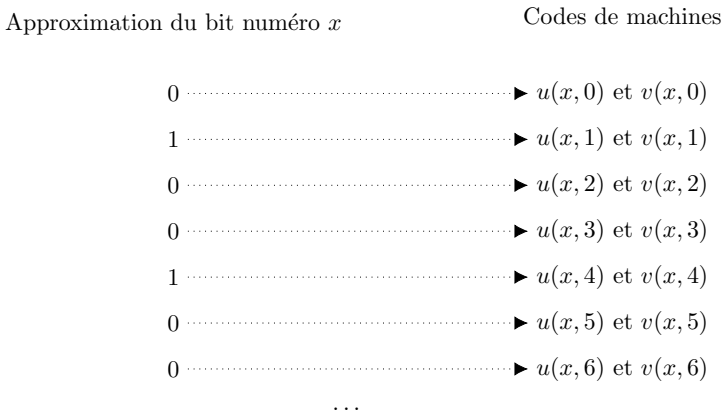



FIGURE 7.3 – *Illustration de la preuve de $\Leftarrow$ du lemme de Shoenfield : étant donné l'approximation d'un bit, on crée à l'étape $y$ le code $u(x,y)$ du programme qui s'arrête partout si une valeur différente de 1 est prise par le bit $x$ à une étape plus grande que $y$, et le code $v(x,y)$ du programme qui s'arrête partout si une valeur différente de 0 est prise par le bit $x$ à une étape plus grande que $y$.*

**Exercice 7.5.** Montrer que si $Y$ est $X$-c. e. et $Z$ est $Y$-c. e., alors $Z$ n'est pas nécessairement $X$-c. e. ◇

Tout ensemble $\emptyset'$-calculable $A$ se voit donc associer une fonction stable $f$ dont la limite est $A$. On présente souvent cette fonction sous la forme d'une succession d'ensembles *uniformément calculables* $A_0, A_1, \ldots$ définie par $A_y = \{x : f(x,y) = 1\}$ pour tout $y$.

**La calculabilité uniforme**

Nous avons introduit le concept de suite $A_0, A_1, \ldots$ *uniformément calculable* : chaque élément $A_n$ de la suite est calculable par *la même fonction*, paramétrée par un paramètre supplémentaire $n$ qui indique que l'on calcule le $n$-ième élément de la suite.
Insistons sur le fait qu'une suite d'ensembles calculables $(X_i)_{i\in\mathbb{N}}$ n'est pas nécessairement uniformément calculable : il n'existe pas forcément d'algorithme permettant de calculer $X_i$ en fonction de $i$. À titre d'exemple trivial, chaque $X_i$ peut simplement être une suite infinie de 0, à l'exception du bit en position $i$ qui est égal au $i$-ième bit de $\emptyset'$. Chaque $X_i$ est un ensemble fini et est donc de ce fait calculable. En revanche, un algorithme permettant de calculer *uniformément* $X_i$ en fonction de $i$ permettrait de calculer l'arrêt, et ne peut donc exister.

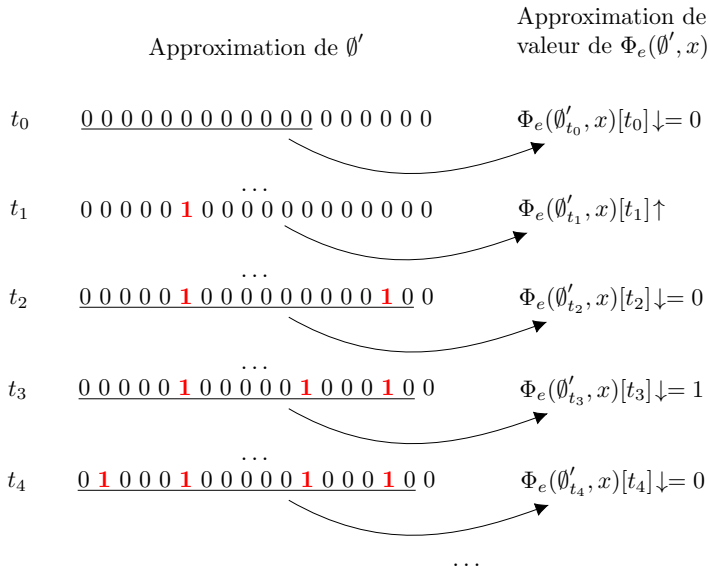



FIGURE 7.4 – *Illustration de la preuve de $\Rightarrow$ du lemme de Shoenfield. La première colonne représente des approximations successives de $\emptyset'$. La partie soulignée représente l'usage du calcul $\Phi_e(\emptyset', x)[t_n]$ lorsqu'il s'arrête. À mesure que l'on énumère le $i$-ième élément dans l'arrêt à l'étape de calcul $t_i$, on lance le calcul de $\Phi_e$ avec l'approximation courante de l'arrêt comme oracle, et pour $t_i$ étapes de calcul. Par la propriété de l'usage, le processus converge nécessairement.*

**Définition 7.6.** Soit $A \leqslant_T \emptyset'$ un ensemble. Une *approximation* $\Delta^0_2$ de $A$ est une suite uniformément calculable d'ensembles $A_0, A_1, \ldots$ telle que pour tout $x$, $\lim_y A_y(x)$ existe et vaut $A(x)$. ♢

Les approximations $\Delta^0_2$ ne sont pas canoniques. Il est toujours possible par exemple d'« accélérer » une approximation $\Delta^0_2$ $A_0, A_1, \ldots$ en considérant la suite $A_{g(0)}, A_{g(1)}, \ldots$ pour une fonction calculable et strictement croissante $g : \mathbb{N} \to \mathbb{N}$.

**Remarque**

L'appellation « approximation $\Delta^0_2$ » sera justifiée dans le chapitre 5, où l'on donnera une nouvelle caractérisation des ensembles $\emptyset'$-calculables comme ceux définissables par un prédicat $\Delta^0_2$.

Les approximations $\Delta^0_2$ permettent de définir deux fonctions importantes, à savoir le modulus et la fonction de calcul. Ces fonctions expriment la com-

plexité calculatoire de l'ensemble $A$ sous forme de rapidité de croissance : toute fonction croissant plus vite que le modulus de $A$ ou que sa fonction de calcul permet de recalculer $A$.

**Définition 7.7.** Soit $A_0, A_1, A_2, \ldots$ une approximation $\Delta_2^0$ d'un ensemble $A$.

1. Le *modulus* de l'approximation $\Delta_2^0$ est la fonction $\mu_A : \mathbb{N} \to \mathbb{N}$ qui à $x$ associe le plus petit entier $n$ tel que la suite $A_n \restriction_x, A_{n+1} \restriction_x, \ldots$ soit constante.
2. La *fonction de calcul* de l'approximation $\Delta_2^0$ est la fonction $c_A : \mathbb{N} \to \mathbb{N}$ qui à $x$ associe le plus petit entier $n \geqslant x$ tel que $A_n \restriction_x = A \restriction_x$. ♢

Contrairement aux approximations c.e qui, lorsqu'elles font apparaître un élément dans $A$, ne le retirent plus jamais, une approximation $\Delta_2^0$ de $A$ a le droit de « changer d'avis » un nombre arbitrairement grand (mais fini) de fois sur l'appartenance d'un élément $x$ à $A$. En particulier, la fonction de calcul croît en général plus lentement que le modulus, car un ensemble $A_n$ peut coïncider avec l'ensemble $A$ sur un long segment initial sans pour autant avoir atteint son seuil de stabilité sur ce segment. La fonction de calcul, contrairement au modulus en général, est calculable par l'ensemble $A$.

Il est facile de vérifier que toute fonction dominant le modulus d'une approximation $\Delta_2^0$ d'un ensemble calcule cet ensemble.

**Exercice 7.8.** Soit $A_0, A_1, \ldots$ une approximation $\Delta_2^0$ d'un ensemble $A$, et soit $\mu_A$ son modulus. Montrer que toute fonction dominant $\mu_A$ calcule $A$. Une fonction $f$ domine une fonction $g$ si $f(n) \geqslant g(n)$ pour tout $n \in \mathbb{N}$. ⋄

Il n'est cependant pas clair que cela soit également le cas de la fonction de calcul. C'est pourtant ce que montre la proposition suivante.

**Proposition 7.9 (Martin et Miller [157]).** Soit $A_0, A_1, \ldots$ une approximation $\Delta_2^0$ d'un ensemble $A$. Soit $c_A$ sa fonction de calcul. Toute fonction dominant $c_A$ calcule $A$. ⋆

Preuve. Soit $f$ une fonction dominant $c_A$. Soit $M(x)$ le plus grand $y \leqslant x$ tel que pour tout $x \leqslant t \leqslant f(x)$, $A_t \restriction_y = A_{f(x)} \restriction_y$. La fonction $M$ est totale $f$-calculable. De plus, $M$ tend vers $+\infty$, car l'approximation de $A$ étant $\Delta_2^0$, elle va se stabiliser sur des segments initiaux de plus en plus grands. Enfin, comme $x \leqslant c_A(x) \leqslant f(x)$, alors si $M(x) = y$, $A_x \restriction_y = A_{c_A(x)} \restriction_y = A \restriction_y$. Ainsi, pour décider si $n \in A$, il suffit de trouver un entier $x$ tel que $M(x) > n$, puis tester si $n \in A_x$. Cette procédure est $f$-calculable. ∎

Notons qu'il est important de demander que $c_A(x) \geqslant x$ dans la définition de fonction de calcul. La proposition précédente devient fausse lorsque l'on omet cette précision.

**Remarque**

Les notions de modulus et de fonction de calcul ne sont pas caractéristiques d'un ensemble $\emptyset'$-calculable, mais d'une approximation $\Delta^0_2$ d'un ensemble.
Un même ensemble $\emptyset'$-calculable possède une infinité d'approximations $\Delta^0_2$, chacune ayant son modulus et sa fonction de calcul.

## 8. Méthode des extensions finies

Nous allons maintenant présenter une méthode relativement simple et pourtant très puissante, permettant de créer des ensembles satisfaisant des propriétés calculatoires « sur mesure ». Il s'agit de la *méthode des extensions finies*.

En calculabilité, on appelle *propriété de faiblesse* une propriété close par le bas dans les degrés Turing, c'est-à-dire que si $X$ a une propriété de faiblesse et si $Y \leqslant_T X$, alors $Y$ a aussi cette propriété de faiblesse. À l'inverse, une *propriété de force* est une propriété close par le haut dans les degrés Turing, c'est-à-dire que si $X$ a une propriété de force et si $X \leqslant_T Y$, alors $Y$ a aussi cette propriété de force.

La méthode des extensions finies est particulièrement adaptée lorsque l'on se pose la question de l'existence d'ensembles satisfaisant simultanément des propriétés de force et de faiblesse. Il faut alors créer un ensemble sur mesure, ni trop fort, ni trop faible du point de vue calculatoire. Nous allons illustrer cette méthode en prouvant deux propositions.

**Proposition 8.1 (Kleene et Post, 1954).** Il existe deux ensembles $A$ et $B$ incomparables par la réduction de Turing. ⋆

Preuve. Nous allons construire simultanément deux ensembles $A$ et $B$, vus comme des suites binaires infinies — souvenons-nous de la correspondance entre les deux.

L'ensemble $A$ doit satisfaire une propriété de force ($A$ n'est pas calculable par $B$) et une propriété de faiblesse ($A$ ne calcule pas $B$). L'ensemble $B$ doit de son côté satisfaire les propriétés duales de force et de faiblesse.

**Contrats**. Les propriétés calculatoires, qu'elles soient de force ou de faiblesse, sont en général des schémas de propriétés, au sens où elles se déclinent en une infinité de propriétés plus élémentaires et plus faciles à satisfaire de manière indépendante. Par exemple, la propriété « $A \not\geqslant_T B$ » correspond à la collection de propriétés « $\Phi_e^A \neq B$ » pour tout code de fonctionnelle $e$, où « $\Phi_e^A \neq B$ » signifie que soit $\Phi_e^A$ est une fonction partielle, soit $\Phi_e^A(x) \downarrow \neq B(x)$ pour un $x \in \mathbb{N}$. On appelle ces propriétés

élémentaires des *contrats* (ou *requirements* en anglais). La première étape d'une construction par méthode des extensions finies consiste à identifier les contrats. Nous en avons donc de deux sortes $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ :

$$\begin{aligned} \mathcal{R}_e &: \exists x\ \Phi_e^A(x)\uparrow\ \vee\ \exists x\ \Phi_e^A(x)\downarrow\neq B(x) \\ \mathcal{S}_e &: \exists x\ \Phi_e^B(x)\uparrow\ \vee\ \exists x\ \Phi_e^B(x)\downarrow\neq A(x). \end{aligned}$$

Si tous les contrats $\mathcal{R}_e$ sont satisfaits, alors $A \not\geqslant_T B$, tandis que si tous les contrats $\mathcal{S}_e$ le sont, $B \not\geqslant_T A$. Étant donné que l'on ne s'intéresse qu'aux fonctionnelles calculant des éléments de $2^{\mathbb{N}}$, on considérera — comme souvent dans ce genre de cas — que $\Phi_e(X,n)\uparrow$ si jamais $\Phi_e(X,n)\downarrow\notin\{0,1\}$.

**Satisfaction d'un contrat**. Supposons que l'on ne veuille satisfaire qu'un seul contrat, disons $\mathcal{R}_e$. Deux cas se présentent.

- ▷ Cas 1. Il existe un ensemble $X$ tel que $\Phi_e^X(0)\downarrow= i$ pour un $i\in\{0,1\}$ donné. On peut alors fixer $A = X$ et $B$ peut être n'importe quel ensemble tel que $B(0)\neq i$. On aura alors satisfait le contrat $\mathcal{R}_e$ en s'assurant que $\Phi_e^A(0)\downarrow\neq B(0)$.
- ▷ Cas 2. Quel que soit l'ensemble $X$, on a $\Phi_e^X(0)\uparrow$. Ce cas-ci est encore plus simple, $A$ et $B$ peuvent être n'importe quels ensembles.

L'argument précédent a entièrement spécifié les ensembles $A$ et $B$ pour satisfaire un contrat $\mathcal{R}_e$, sans laisser de liberté aux autres contrats pour être satisfaits. Nous allons donc tenter d'être plus économes pour laisser de la place à la satisfaction des autres contrats. Pour cela, nous allons faire en sorte de ne spécifier qu'un segment initial fini de $A$ et $B$ pour satisfaire un contrat donné.

**Construction**. Les ensembles $A$ et $B$ vont être construits par approximations finies, sous forme de deux suites de chaînes binaires finies, représentant des préfixes de plus en plus longs de $A$ et $B$.

$$\sigma_0 \preceq \sigma_1 \preceq \dots \qquad \text{et} \qquad \tau_0 \preceq \tau_1 \preceq \dots$$

On ne demande pas nécessairement à ce que $\sigma_n \prec \sigma_{n+1}$ : la suite de chaînes peut stagner pendant un certain temps. En revanche, pour tout $n$, on demande que pour $m > n$ suffisamment grand on ait $\sigma_n \prec \sigma_m$. De cette manière, les chaînes $\sigma_0 \preceq \sigma_1 \preceq \dots$ convergent petit à petit vers une unique suite infinie $A$ et les chaînes $\tau_0 \preceq \tau_1 \preceq \dots$ convergent petit à petit vers une unique suite infinie $B$. Formellement, on définit $A$ et $B$ via de nouvelles notations : étant donné une chaîne binaire $\sigma$, on notera $[\sigma]$ l'ensemble des suites binaires infinies ayant $\sigma$ comme préfixe. On aura donc $A\in[\sigma_n]$ et $B\in[\tau_n]$ pour tout $n\in\mathbb{N}$. En s'assurant qu'il existe des chaînes de longueur arbitrairement grande, on fait en sorte que $\bigcap_n[\sigma_n]$ et $\bigcap_n[\tau_n]$ contiennent chacun exactement un élément : $A$ et $B$ respectivement.

Les approximations finies de $A$ et $B$ vont être définies par étapes, de manière à satisfaire successivement chaque contrat. Comme à une étape donnée, seuls des préfixes finis de $A$ et $B$ sont connus, il va donc falloir satisfaire un contrat quel que soit ce qui viendra après ces préfixes dans la suite de la construction. Une paire de chaînes $\sigma_n$ et $\tau_n$ *force* un contrat $\mathcal{R}_e$ (ou un contrat $\mathcal{S}_e$) si la propriété du contrat est satisfaite pour tous $A \succeq \sigma_n$ et $B \succeq \tau_n$. On doit donc s'assurer que le contrat est satisfait pour tous les éléments de $[\sigma_n]$ et de $[\tau_n]$. Notons au passage que si $\sigma_n$ et $\tau_n$ forcent un contrat, alors pour tout $\sigma \succeq \sigma_n$ et $\tau \succeq \tau_n$ on a $[\sigma] \subseteq [\sigma_n]$ et $[\tau] \subseteq [\tau_n]$, et donc $\sigma$ et $\tau$ forcent encore le contrat.

Les différents contrats vont être entrelacés afin que chacun reçoive l'attention à une étape de la construction. Lors d'une étape paire $n = 2e$, on définira $\sigma_{n+1}$ et $\tau_{n+1}$ de manière à forcer $\mathcal{R}_e$, tandis que lors d'une étape impaire, $n = 2e+1$, on forcera $\mathcal{S}_e$. Les contrats seront donc satisfaits selon l'ordre

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \ldots$$

**Satisfaction d'un contrat**. Nous allons maintenant à nouveau satisfaire le contrat $\mathcal{R}_e$ en ne spécifiant cette fois-ci qu'un préfixe fini des oracles $A$ et $B$. Le cas des contrats $\mathcal{S}_e$ est symétrique. Supposons que des segments initiaux $\sigma_n$ et $\tau_n$ ont déjà été spécifiés pour $A$ et $B$ (on commence au départ la construction avec $\sigma_0 = \tau_0 = \epsilon$ où $\epsilon$ est le mot vide). Autrement dit, les suites binaires infinies finales $A$ et $B$ devront respecter $\sigma_n \preceq A$ et $\tau_n \preceq B$. Soit $x = |\tau_n|$. En particulier, $x$ est la première position sur laquelle $B$ n'est pas encore spécifié. Toutes les valeurs de $B$ aux positions précédant $x$ sont déjà fixées par $\tau_n$. Les deux cas suivants se présentent.

- ▷ Cas 1. Il existe un ensemble $X \succeq \sigma_n$, tel que $\Phi_e^X(x)\!\downarrow = i$ pour un $i$ dans $\{0,1\}$ donné. Dans ce cas, par la propriété de l'usage, ce calcul fait appel à un nombre fini de bits de l'oracle, et il existe donc un segment initial $\sigma_{n+1} \preceq X$ tel que $\Phi_e^{\sigma_{n+1}}(x)\!\downarrow = i$, ou autrement dit tel que $\Phi_e^Y(x)\!\downarrow = i$ pour tout ensemble $Y \succeq \sigma_{n+1}$. On peut choisir le segment initial $\sigma_{n+1}$ de manière à ce qu'il soit au moins aussi long que $\sigma_n$, ce qui assure $\sigma_{n+1} \succeq \sigma_n$. Sachant que l'ensemble $A$ aura $\sigma_{n+1}$ pour segment initial, on s'est assuré que $\Phi_e^A(x)\!\downarrow = i$. Soit $\tau_{n+1}$ la chaîne obtenue à partir de $\tau_n$ en lui ajoutant le bit $1-i$. Autrement dit, $|\tau_{n+1}| = |\tau_n| + 1$, $\tau_{n+1} \succeq \tau_n$ et $\tau_{n+1}(x) = 1 - i$. On s'est alors assuré que pour tout $B \succeq \tau_{n+1}$, $B(x) = 1 - i$. Ainsi, les chaînes $\sigma_{n+1}$ et $\tau_{n+1}$ étendent respectivement $\sigma_n$ et $\tau_n$, et forcent le contrat $\mathcal{R}_e$ en s'assurant que $\Phi_e^A(x)\!\downarrow \neq B(x)$ pour tous $A \succeq \sigma_n$ et $B \succeq \tau_n$.
- ▷ Cas 2. Pour tout ensemble $X \succeq \sigma_n$, on a $\Phi_e^X(x)\!\uparrow$. Dans ce cas, $\sigma_n$ et $\tau_n$ forcent déjà le contrat $\mathcal{R}_e$ en s'assurant $\Phi_e^A(x)\!\uparrow$ pour un certain $x$. Il suffit donc de prendre $\sigma_{n+1} = \sigma_n$ et $\tau_{n+1} = \tau_n$.

On s'assurera que les longueurs des éléments des suites $(\sigma_n)_{n\in\mathbb{N}}$ et $(\tau_n)_{n\in\mathbb{N}}$ soient de plus en plus grandes, en ajoutant un bit arbitraire au bout de chaque chaîne à la fin de chaque étape, avant de passer à l'étape suivante. Comme expliqué précédemment, le fait de forcer un contrat est clos par extension de chaînes, et cela n'altérera donc pas la validité de la construction. La preuve de la proposition 8.1 est terminée. ∎

Une nouvelle notation a été introduite dans la preuve précédente ; nous en officialisons ci-après l'utilisation.

**Notation**

Étant donné $\sigma \in 2^{<\mathbb{N}}$, on note $[\sigma]$ l'ensemble des $X \in 2^{\mathbb{N}}$ tels que $\sigma \prec X$.

Nous allons donner une autre illustration de la méthode des extensions finies en prouvant une proposition plus forte, impliquant la proposition 8.1. On fixe cette fois-ci un ensemble $A$, arbitraire en dehors du fait qu'on le suppose non calculable. On construit alors un ensemble $B$ que $A$ ne calcule pas, et qui ne calcule pas $A$. Il s'agit d'une construction *a priori* plus difficile, car on ne contrôle plus qu'un seul des deux ensembles. Notons par ailleurs qu'il sera nécessaire pour cette preuve d'utiliser le fait que $A$ est non calculable : en effet, dans le cas inverse, tout ensemble $B$ que l'on construit permettrait de calculer $A$.

**Proposition 8.2.** Pour tout ensemble non calculable $A$, il existe un ensemble $B$ tel que $B \not\leqslant_T A$ et $A \not\leqslant_T B$. ⋆

Preuve. Contrairement à la proposition 8.1, l'ensemble $A$ est déjà fixé. Nous allons construire uniquement l'ensemble $B$ par la méthode des approximations finies. L'ensemble $B$ doit encore satisfaire une propriété de force ($B \not\leqslant_T A$) et une propriété de faiblesse ($A \not\leqslant_T B$). Les contrats sont donc identiques à ceux de la proposition 8.1, à savoir

$$\begin{aligned} \mathcal{R}_e &: \exists x \Phi_e^A(x)\uparrow \vee \exists x \Phi_e^A(x)\downarrow \neq B(x) \\ \mathcal{S}_e &: \exists x \Phi_e^B(x)\uparrow \vee \exists x \Phi_e^B(x)\downarrow \neq A(x). \end{aligned}$$

Cependant, les ensembles $A$ et $B$ ne jouant plus un rôle symétrique, les contrats $\mathcal{R}_e$ et $\mathcal{S}_e$ vont être satisfaits chacun à leur manière.

**Construction**. L'ensemble $B$ va être construit par approximations successives

$$\tau_0 \preceq \tau_1 \preceq \tau_2 \preceq \dots$$

pour définir $B$ comme unique élément de l'ensemble $\bigcap_n [\tau_n]$. Une chaîne $\tau_n$ *force* un contrat $\mathcal{R}_e$ (ou un contrat $\mathcal{S}_e$) si la propriété est satisfaite pour tout $X \in [\tau_n]$. À chaque étape de la construction, un contrat va être forcé,

en les entrelaçant comme précédemment :

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \ldots$$

**Satisfaction d'un contrat $\mathcal{R}_e$.** À l'étape $n$, supposons que la chaîne $\tau_n$ est définie. Nous voulons trouver une extension $\tau_{n+1} \succeq \tau_n$ forçant le contrat $\mathcal{R}_e$. La satisfaction de ce type de contrat est très similaire à celle de la proposition 8.1. Soit $x = |\tau_n|$. Deux cas se présentent.

- ▷ Cas 1. On a $\Phi_e^A(x)\downarrow = i$ pour un $i \in \{0, 1\}$. Il suffit alors de définir $\tau_{n+1}$ comme l'unique chaîne de longueur $|\tau_n| + 1$ étendant $\tau_n$ telle que
$$\tau_{n+1}(x) = 1 - i.$$
Comme $B \in [\tau_{n+1}]$, $B(x) = 1 - i$, donc $\Phi_e^A(x)\downarrow \neq B(x)$.
- ▷ Cas 2. On a $\Phi_e^A(x)\uparrow$. Le contrat $\mathcal{R}_e$ est alors trivialement satisfait, car $\Phi_e^A$ est une fonction partielle. Il suffit donc de prendre $\tau_{n+1} = \tau_n$.

On notera que l'on n'a fait aucune supposition sur l'ensemble $A$ pour satisfaire les contrats $\mathcal{R}_e$. L'hypothèse selon laquelle $A$ n'est pas calculable sera exploitée pour satisfaire les contrats $\mathcal{S}_e$.

**Satisfaction d'un contrat $\mathcal{S}_e$.** La difficulté de la satisfaction d'un contrat tel que $\mathcal{S}_e$ provient du fait que l'on n'a pas de contrôle sur l'ensemble $A$ qui est entièrement spécifié. Plus précisément, lors de la satisfaction d'un contrat $\mathcal{R}_e$, on fixe une entrée $x$ qui n'est pas encore spécifiée pour $B$, de manière à choisir sa valeur dans le cas 1 pour la rendre différente de la valeur de $\Phi_e^A(x)$. Ce n'est pas possible dans le cas du contrat $\mathcal{S}_e$, toutes les valeurs de $A$ étant fixées. Il va donc falloir exploiter le fait que l'ensemble $A$ n'est pas calculable. Notons en revanche que la satisfaction des contrats $\mathcal{R}_e$ laissait une certaine liberté, notamment dans le choix de $x$. En effet, seul un nombre fini d'entrées est spécifié à une étape donnée pour $B$, et donc la quasi-totalité des entrées peut être choisie pour $x$. Nous allons exploiter cette liberté de choix pour satisfaire les contrats $\mathcal{S}_e$. Trois cas se présentent.

- ▷ Cas 1. Il existe une entrée $x$ et un ensemble $X \succeq \tau_n$ tels que
$$\Phi_e^X(x)\downarrow \neq A(x).$$
Par la propriété de l'usage, il existe alors une chaîne finie $\tau_{n+1} \succeq \tau_n$ telle que $\Phi_e^X(x)\downarrow \neq A(x)$ pour tout $X \in [\tau_{n+1}]$. La chaîne $\tau_{n+1}$ force donc le contrat $\mathcal{S}_e$.
- ▷ Cas 2. Il existe une entrée $x$ telle que pour tous les ensembles $X \succeq \tau_n$, on a $\Phi_e^X(x)\uparrow$. Dans ce cas, la chaîne $\tau_n$ force déjà le contrat $\mathcal{S}_e$ en s'assurant que $\Phi_e^B(x)\uparrow$.
- ▷ Cas 3. Aucun des deux cas précédents ne se présente. Nous allons montrer alors qu'il est possible de calculer l'ensemble $A$, et donc d'en déduire une contradiction.

Voici la procédure pour calculer la valeur de $A(x)$ : chercher une chaîne finie $\tau \succeq \tau_n$ telle que $\Phi_e^\tau(x)\downarrow$ et renvoyer le résultat de ce calcul. Nous prétendons les deux faits suivants :

(1) il existe une telle chaîne, et donc que la recherche se terminera ;

(2) quelle que soit $\tau$ telle que $\Phi_e^\tau(x)\downarrow$, alors $\Phi_e^\tau(x)\downarrow = A(x)$.

Pour montrer (1), remarquons que la négation du cas 2 signifie que pour tout $x$ (et en particulier pour ce $x$ considéré), il existe un ensemble $X \succeq \tau_n$ tel que $\Phi_e^X(x)\downarrow$. Par la propriété de l'usage, il existe alors un segment initial $\tau \succeq \tau_n$ de $X$ tel que $\Phi_e^\tau(x)\downarrow$.
Montrons (2). Si $\Phi_e^\tau(x)\downarrow \neq A(x)$, alors le cas 1 serait vrai en prenant n'importe quel $X \succeq \tau$. Il s'ensuit que $\Phi_e^\tau(x)\downarrow$ implique $\Phi_e^\tau(x) = A(x)$. Nous avons donc décrit une procédure calculable pour déterminer la valeur de $A(x)$ quel que soit $x$, contredisant l'hypothèse selon laquelle $A$ n'est pas calculable.

Cela conclut la preuve de la proposition 8.2. ∎

Avant de conclure cette section dédiée à la méthode des extensions finies, mentionnons que de manière générale, cette méthode n'est pas *effective*, au sens où aucune contrainte de calculabilité n'est imposée à la suite des chaînes finies en construction. Il est cependant possible de faire une analyse fine de l'argument pour déterminer la puissance calculatoire nécessaire pour trouver un $\tau_{n+1}$, étant donné $\tau_n$. On obtient alors des bornes supérieures sur la complexité de l'ensemble construit.

# 9. Degrés low

Comme nous l'avons vu, le saut Turing est invariant par degré Turing. Ainsi, pour tout ensemble calculable $X$, $X' \equiv_T \emptyset'$. Il est naturel de se demander si seuls les ensembles calculables ont un saut Turing équivalent à $\emptyset'$, et plus généralement si le saut Turing est une fonction injective sur les degrés Turing. La proposition suivante montre que ce n'est pas le cas.

**Proposition 9.1.** Il existe un ensemble non calculable $A$ tel que

$$A' \equiv_T \emptyset'.$$ ⋆

Preuve. L'ensemble $A$ va être construit à l'aide d'une version effective de la méthode des extensions finies (voir la section 8), en s'assurant que l'intégralité de la construction est calculable en $\emptyset'$.

**Contrats**. L'ensemble $A$ doit satisfaire une propriété de force ($A$ non calculable) et une propriété de faiblesse ($A' \leqslant_T \emptyset'$).

La propriété de force se décline en une infinité de contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{R}_e : \exists x \Phi_e(x)\uparrow \ \vee \exists x \Phi_e(x)\downarrow \neq A(x).$$

La propriété de faiblesse est d'un type nouveau. Pour la satisfaire, nous allons faire en sorte de « contrôler » le saut Turing de $A$ au fur et à mesure de la construction, tout en s'assurant que toute la construction elle-même est calculable en $\emptyset'$. En s'aidant de ce fait, on aura une fonction $\emptyset'$-calculable $f$ pour laquelle on devra satisfaire une infinité de contrats $(\mathcal{S}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{S}_e : \ \ \Phi_e^A(e)\downarrow \to f(e) = 1 \ \text{ et } \ \Phi_e^A(e)\uparrow \to f(e) = 0.$$

Informellement, le contrat $\mathcal{S}_e$ est satisfait si, à un moment fini de la construction, nous savons si $\Phi_e^A(e)\downarrow$ ou $\Phi_e^A(e)\uparrow$, quelle que soit la suite de la construction.

**Construction**. L'ensemble $A$ va être construit par approximations successives

$$\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$$

pour définir $A$ comme unique élément de $\bigcap_n [\sigma_n]$. Une chaîne $\sigma_n$ *force* un contrat $\mathcal{R}_e$ ou $\mathcal{S}_e$ si la propriété est satisfaite pour tout $B \in [\sigma_n]$. À chaque étape de la construction, un contrat va être forcé, en les entrelaçant comme précédemment :

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \dots$$

Nous devons de plus nous assurer que la construction est calculable en $\emptyset'$. Ainsi, pour connaître la valeur de $A'(e)$, il suffira d'exécuter à l'aide de $\emptyset'$ la construction jusqu'à l'étape $2e$ satisfaisant $\mathcal{S}_e$, et de renvoyer le résultat. Cette procédure $\emptyset'$-calculable assure que $A' \leqslant_T \emptyset'$. Nous allons donc montrer comment satisfaire chaque type de contrat indépendamment, tout en analysant la complexité calculatoire de chaque étape pour s'assurer que $\sigma_{n+1}$ peut être obtenu à partir de $\sigma_n$ à l'aide de l'oracle $\emptyset'$.

**Satisfaction d'un contrat $\mathcal{R}_e$.** Supposons que $\sigma_n$ est déjà défini. Nous voulons trouver $\sigma_{n+1} \succeq \sigma_n$ forçant le contrat $\mathcal{R}_e$. Soit $x = |\sigma_n|$ ; autrement dit, $x$ est la première valeur de $A$ qui n'est pas encore spécifiée. Deux cas se présentent.

- ▷ Cas 1. On a $\Phi_e(x)\uparrow$, auquel cas $\mathcal{R}_e$ est trivialement satisfait, et l'on peut définir $\sigma_{n+1} = \sigma_n$
- ▷ Cas 2. On a $\Phi_e(x)\downarrow$, auquel cas la chaîne $\sigma_{n+1}$ obtenue à partir de $\sigma_n$ en lui ajoutant le bit $1 - \Phi_e(x)$ force $\mathcal{R}_e$.

En fonction du cas, nous allons donc définir une extension $\sigma_{n+1}$ différente. Il faut s'assurer que cette extension peut être obtenue calculatoirement à l'aide de l'oracle $\emptyset'$. La distinction entre les deux cas n'est pas calculable en soi, car elle demande de décider si $\Phi_e(x)$ s'arrête ou non. Nous pouvons

cependant utiliser $\emptyset'$ pour répondre à cette question comme suit : soit $\Phi_i$ la fonction partielle calculable définie pour tout $n$ par $\Phi_i(n) = \Phi_e(x)$. Le code $i$ peut être construit calculatoirement, car il s'agit d'une simple manipulation de machine. Il s'ensuit que l'on est dans le premier cas ssi $i \notin \emptyset'$. Dans le premier cas, $\sigma_{n+1} = \sigma_n$ est trivialement calculable, tandis que dans le second cas, il suffit d'exécuter $\Phi_e(x)$ pour récupérer la valeur renvoyée, et obtenir ainsi $\sigma_{n+1}$. Nous pouvons donc trouver une extension $\sigma_{n+1}$ forçant le contrat $\mathcal{R}_e$ à l'aide de l'oracle $\emptyset'$.

**Satisfaction d'un contrat $\mathcal{S}_e$.** Supposons que $\sigma_n$ est déjà défini. Nous voulons trouver $\sigma_{n+1} \succeq \sigma_n$ tel que le comportement de $\Phi_e^A(e)$ est déjà défini par $\sigma_{n+1}$, autrement dit $\Phi_e^X(e)\downarrow$ pour tout $X \in [\sigma_{n+1}]$ ou $\Phi_e^X(e)\uparrow$ pour tout $X \in [\sigma_{n+1}]$. Deux cas se présentent encore.

- ▷ Cas 1. Il existe une chaîne $\tau \succeq \sigma_n$ telle que $\Phi_e^\tau(e)\downarrow$. Dans ce cas, en prenant $\sigma_{n+1} = \tau$, nous assurons que $\Phi_e^A(e)\downarrow$, car $A \in [\sigma_{n+1}]$.
- ▷ Cas 2. Pour toute chaîne $\tau \succeq \sigma_n$, $\Phi_e^\tau(e)\uparrow$. Dans ce cas, par la propriété de l'usage, quel que soit l'oracle $A \in [\sigma_n]$, $\Phi_e^A(e)\uparrow$. En définissant $\sigma_{n+1} = \sigma_n$, nous forçons donc $\Phi_e^A(e)\uparrow$.

Ainsi, dans chacun des cas, nous avons forcé le comportement de $\Phi_e^A(e)$ avec un préfixe fini de l'oracle.

Ici encore, il s'agit de trouver $\sigma_{n+1}$ à partir de $\sigma_n$ calculatoirement à l'aide de l'oracle $\emptyset'$. Comme pour le contrat $\mathcal{R}_e$, nous définissons une fonction partielle calculable $\Phi_i$ qui, pour chacune de ses entrées, cherche une chaîne $\tau \succeq \sigma_n$ telle que $\Phi_e^\tau(e)[|\tau|]\downarrow$, et s'arrête s'il en trouve une. Ainsi, $i \in \emptyset'$ si, et seulement si, nous sommes dans le premier cas. Une fois le cas déterminé, la chaîne $\sigma_{n+1}$ peut être trouvée calculatoirement. Cela conclut la preuve de la proposition 9.1. ∎

Parmi les premières notions de faiblesse introduites et étudiées en calculabilité par Cooper et Soare de manière indépendante, se trouve la hiérarchie des ensembles $\text{low}_n$, dont nous donnons ici le premier niveau.

**Définition 9.2.** Un ensemble $A \subseteq \mathbb{N}$ est *low* si $A' \leqslant_T \emptyset'$. ♢

Informellement, un ensemble est low s'il est indistinguable d'un ensemble calculable du point de vue du saut Turing. Nous avons vu que si $X \leqslant_T Y$, alors $X' \leqslant_T Y'$.
Ainsi, comme $\emptyset \leqslant_T A$ pour tout ensemble $A$, $\emptyset' \leqslant_T A'$. Il s'ensuit qu'un ensemble est low ssi $A' \equiv_T \emptyset'$. Nous verrons plus loin d'autres exemples d'ensembles low non calculables, notamment des ensembles calculatoirement énumérables (voir le chapitre 13).

## 10. Degrés high

Si l'on considère un ensemble $A \leqslant_T \emptyset'$, quelles sont les puissances extrêmes que peut prendre son saut Turing $A'$ ?

Rappelons que si $X \leqslant_T Y$, alors $X' \leqslant_T Y'$. En particulier, $A' \geqslant_T \emptyset'$. D'un autre côté, $A' \leqslant_T \emptyset''$ car $A \leqslant_T \emptyset'$. On a donc

$$\emptyset' \leqslant_T A' \leqslant_T \emptyset'' \text{ si } A \leqslant_T \emptyset'.$$

Notons que dans le cas où $A \not\leqslant_T \emptyset'$, le saut Turing de $A$ peut être arbitrairement complexe.

Les ensembles dont le saut Turing est égal à la borne minimale, à savoir $\emptyset'$, sont les ensembles low. Nous avons vu qu'il existait des ensembles low non calculables. Nous allons maintenant nous pencher sur les ensembles dont le saut Turing calcule la borne maximale $\emptyset''$.

**Définition 10.1.** Un ensemble $A \subseteq \mathbb{N}$ est *high* si $\emptyset'' \leqslant_T A'$. ♢

**Remarque**

La notion d'ensemble high a été historiquement définie uniquement pour les ensembles $A \leqslant_T \emptyset'$. Elle a depuis été étendue à tous les ensembles, mais il est important de garder cette différence historique en tête lorsqu'on lit les articles fondateurs de la calculabilité.

Réfléchissons à présent aux ensembles high. À l'inverse des low, leur saut Turing a plus de puissance de calcul qu'attendu : il permet de calculer le double saut. Le problème de l'arrêt lui-même est évidemment un exemple trivial d'ensemble high. Tout comme pour les ensembles low, la notion d'ensemble high a son intérêt pour les exemples non triviaux : les ensembles high qui ne calculent pas $\emptyset'$. Il est en fait possible de montrer que pour tout ensemble $C$ non calculable il existe un ensemble high qui ne calcule pas $C$.

Dans la preuve suivante, nous utilisons pour la première fois des oracles qui sont non pas des ensembles d'entiers, mais des fonctions $f : \mathbb{N} \to \mathbb{N}$. Une telle fonction peut être représentée par la suite binaire infinie $X_f$ telle que $X_f(\langle n, m\rangle) = 1$ ssi $f(n) = m$. Il est clair que $X_f$ permet de calculer $f$, et que tout ensemble $Y$ permettant de calculer $f$ peut aussi calculer $X_f$ : l'ensemble $X_f$ est une représentation minimum de la fonction $f$ dans les degrés Turing. D'autres représentations équivalentes en termes de degré Turing sont possibles, par exemple avec $X_f = 1^{f(0)}01^{f(1)}01^{f(2)}0\ldots$ (On commence par les $f(0)$ premiers bits à 1, suivis d'un 0, puis on continue avec les $f(1)$ bits suivants à 1, suivit d'un 0, etc.).

**Proposition 10.2.** Pour tout ensemble non calculable $C$, il existe un ensemble high $A$ tel que $A \not\geqslant_T C$. ⋆

Preuve. Elle est assez similaire à celle de la proposition 8.2 avec la méthode des extensions finies.

**Contrats**. L'ensemble $A$ doit satisfaire une propriété de force ($A' \geqslant_T \emptyset''$) et une propriété de faiblesse ($C \not\leqslant_T A$). Les contrats pour la propriété de faiblesse sont standard, à savoir

$$\mathcal{S}_e : \exists x \Phi_e^A(x)\uparrow \ \vee \exists x \Phi_e^A(x)\downarrow \neq C(x).$$

Le cas de la propriété de force est différent. Tout d'abord, il ne s'agit pas de contrôler ce que l'ensemble $A$ calcule, mais de contrôler ce que son saut Turing calcule. Ensuite, cette propriété de force ne s'exprime pas sous une forme négative (ne pas se faire calculer par un autre ensemble) mais sous une forme positive (calculer un objet compliqué). Les formulations négatives se prouvent souvent par diagonalisation, tandis que les formulations positives sont plus constructives. Nous n'allons donc pas exprimer la propriété de force sous forme de contrats, mais l'imposer structurellement dans la nature de l'objet que l'on construit.

Dans les utilisations précédentes de la méthode des extensions finies, nous avions construit un ensemble en créant une suite infinie de chaînes binaires formant des approximations finies de l'ensemble.

Cette fois-ci, nous allons construire une fonction stable $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ dont la limite est $\emptyset''$. Comme expliqué dans le paragraphe précédant la preuve, rappelons que cela peut se ramener à l'utilisation d'un élément de $2^{\mathbb{N}}$ en considérant l'ensemble $X_f$ défini par $\langle n, m\rangle \in X_f$ ssi $f(n) = m$. Par la relativisation du lemme de limite de Shoenfield à $f$ (voir le lemme 7.2), un ensemble $B$ est $f$-calculable à la limite ssi $B \leqslant_T f'$. En particulier, pour $B = \emptyset''$, si $\emptyset''$ est $f$-calculable à la limite, alors $\emptyset'' \leqslant_T f'$, autrement dit $f$ est high. L'ensemble $A$ est donc n'importe quel ensemble dans le degré Turing de $f$.

**Construction**. La fonction $f$ va être construite à partir d'approximations finies successives de plus en plus précises. Ces approximations, au lieu d'être des chaînes binaires, vont être des couples $(g, m)$, où

- ▷ $g \subseteq \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ est une fonction partielle à deux paramètres dont le domaine est fini, représentant un morceau de la fonction $f$ que l'on est en train de construire.
- ▷ $m$ est un entier signifiant que désormais, lorsque l'on étendra le domaine de $g$ avec une nouvelle entrée $(x, y)$, si $x < m$ alors $g(x, y) = \emptyset''(x)$.

Autrement dit, la limite des $m$ premières « colonnes » de la fonction $f$ est déjà atteinte et a la bonne valeur. Nous appellerons ces couples des *conditions*, car elles conditionnent une partie du comportement de la fonction $f$.

De la même manière que la relation de suffixe $\sigma \preceq \tau$ pour les chaînes binaires signifie que $\tau$ est une approximation plus précise de la suite que l'on construit, nous allons définir une relation d'extension sur les conditions $(g, m) \preceq (h, n)$ pour signifier que la condition $(h, n)$ est plus précise, ou plus contraignante, que la condition $(g, m)$.

Étant donné une fonction partielle $h \subseteq \mathbb{N} \times \mathbb{N} \to \{0, 1\}$, on notera $\operatorname{dom} h$ son domaine de définition. On dit donc que la condition $(h, n)$ *étend* $(g, m)$ (noté $(h, n) \succeq (g, m)$) si $n \geqslant m$ et si, en outre,

(P1) $g \subseteq h$, c'est-à-dire $\operatorname{dom} g \subseteq \operatorname{dom} h$ et pour tout $(x, y) \in \operatorname{dom} g$,
$$g(x, y) = h(x, y)\,;$$

(P2) pour tout $(x, y) \in \operatorname{dom} h \setminus \operatorname{dom} g$, si $x < m$, alors $h(x, y) = \emptyset''(x)$.

La propriété (P1) signifie que les fonctions finies doivent être compatibles, et plus précisément que la fonction $h$ doit étendre la fonction $g$, tandis que la propriété (P2) formalise l'idée selon laquelle le second paramètre d'une condition fixe les colonnes de la fonction en les stabilisant. La figure 10.3 illustre ce qui vient d'être expliqué.

Arrêtons-nous un instant pour nous familiariser avec ce nouvel objet mathématique et apprendre à le manipuler. Tout d'abord, pour toute condition $(g, m)$ et tout $n$, $(g, n)$ est également une condition. Si de plus $n \geqslant m$, alors $(g, n)$ est une extension. L'entier $m$ n'impose aucune contrainte en soi sur la fonction finie $g$ pour former une condition $(g, m)$. En revanche, $m$ impose des restrictions sur les extensions de $(g, m)$. Plus précisément, si $n \geqslant m$, alors l'ensemble des extensions de $(g, n)$ est un sous-ensemble des extensions de $(g, m)$. Enfin, $(\emptyset, 0)$ est une condition valide, où $\emptyset$ est la fonction définie nulle part.

Du point de vue de la calculabilité, l'ensemble des conditions est un ensemble calculable. En revanche, la relation d'extension entre deux conditions n'est pas calculable à cause de la propriété (P2) qui fait intervenir $\emptyset''$. Cependant, si l'on fixe $m$, alors la relation d'extension entre des conditions ayant $m$ pour seconde composante est calculable, car cela ne fait intervenir qu'un segment fini $\emptyset'' \restriction_m$ de l'ensemble $\emptyset''$. Il suffit de « coder en dur » ce segment initial dans le programme. Cette observation sera exploitée pour satisfaire les contrats $\mathcal{S}_e$.

De la même manière que l'on note $[\sigma]$ l'ensemble des suites binaires infinies ayant pour segment initial $\sigma$, on notera $[g, m]$ l'ensemble des fonctions totales $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ telles que $g \subseteq f$ et telles que pour

| | | | | $m = 3$ | | | |
|---|---|---|---|---|---|---|---|
| $y$ | $f(0,y)$ | $f(1,y)$ | $f(2,y)$ | $f(3,y)$ | $f(4,y)$ | $f(5,y)$ | $f(6,y)$ |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 6 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 7 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| | = | = | = | = | = | = | = |
| | $\emptyset''(0)$ | $\emptyset''(1)$ | $\emptyset''(2)$ | $\emptyset''(3)$ | $\emptyset''(4)$ | $\emptyset''(5)$ | $\emptyset''(6)$ |

FIGURE 10.3 – *La partie en gris foncé représente une condition avec $m = 3$. La partie en gris clair représente une extension de cette condition : chaque $i$-ième colonne pour $i < m$ est fixée pour toute extension à une valeur qui doit correspondre au $i$-ième bit du double arrêt.*

tout $(x, y) \notin \operatorname{dom} g$ avec $x < m$ on a $f(x, y) = \emptyset''(x)$. Ainsi, $[g, m]$ est la collection de l'ensemble des fonctions candidates que l'on peut obtenir en complétant l'approximation partielle $(g, m)$. Notons que $[g, m]$ ne contient pas que des fonctions stables.

Nous allons donc construire $f$ par approximations successives sous forme de conditions

$$(g_0, m_0) \preceq (g_1, m_1) \preceq (g_2, m_2) \preceq \dots$$

pour définir $f = \bigcup_t g_t$. Autrement dit, $\operatorname{dom} f = \bigcup_t \operatorname{dom} g_t$, et pour tout couple $(x, y) \in \operatorname{dom} f$, $f(x, y) = g_t(x, y)$ pour un $t$ tel que $(x, y) \in \operatorname{dom} g_t$. La fonction $f$ est bien définie grâce à la propriété de compatibilité (P1). Si l'on s'assure que les entiers $m_t$ deviennent arbitrairement grands, il est facile de vérifier par la propriété (P2) que la fonction $f$ résultante est stable, et a pour limite $\emptyset''$. Notons en particulier que $f \in \bigcap_t [g_t, m_t]$.

Une condition $(g, m)$ *force* un contrat $\mathcal{S}_e$ si la propriété est satisfaite pour toute $f \in [g, m]$. Ici, nous avons remplacé les occurrences de $A$ par $f$ dans le contrat $\mathcal{S}_e$. À chaque étape de la construction, un contrat va être forcé.

**Satisfaction d'un contrat $\mathcal{S}_e$.** L'argument est similaire à celui de la proposition 8.2, mais en manipulant des conditions et non des chaînes binaires. Soit $(g_t, m_t)$ une condition. Les trois cas suivants se présentent.

▷ Cas 1. Il existe une entrée $x$ et une fonction $f$ dans $[g_t, m_t]$ telles que
$$\Phi_e^f(x)\downarrow \neq C(x).$$
Dans ce cas, par la propriété de l'usage, il existe une extension $(g_{t+1}, m_t)$ de $(g_t, m_t)$ telle que $\Phi_e^f(x)\downarrow \neq C(x)$ pour toute $f$ dans $[g_{t+1}, m_t]$. Notons que $m_{t+1} = m_t$. La condition $(g_{t+1}, m_t)$ force donc le contrat $\mathcal{S}_e$.

▷ Cas 2. Il existe une entrée $x$ telle que pour toutes les fonctions $f$ dans $[g_t, m_t]$, $\Phi_e^f(x)\uparrow$. Dans ce cas, la condition $(g_t, m_t)$ force déjà le contrat $\mathcal{S}_e$ en s'assurant que $\Phi_e^f(x)\uparrow$.

▷ Cas 3. Aucun des deux cas précédents ne se présente. Nous allons montrer alors qu'il est possible de calculer l'ensemble $C$, et donc d'en déduire une contradiction. Voici la procédure pour calculer la valeur de $C(x)$ : chercher une condition $(h, m_t)$ étendant $(g_t, m_t)$ avec la même seconde composante $m_t$, telle que $\Phi_e^h(x)\downarrow$. Nous prétendons les deux faits suivants :

(1) il existe une telle extension, et donc que la recherche se terminera ;

(2) quelle que soit $(h, m_t) \succeq (g_t, m_t)$, $\Phi_e^h(x)\downarrow \to \Phi_e^h(x) = C(x)$.

Pour montrer (1), remarquons que la négation du cas 2 signifie que pour tout $x$ (et en particulier pour ce $x$ considéré), il existe une fonction $f \in [g_t, m_t]$ telle que $\Phi_e^f(x)\downarrow$. Par la propriété de l'usage, il existe alors une condition $(h, m_t) \succeq (g_t, m_t)$ telle que $\Phi_e^h(x)\downarrow$.
Montrons (2). Si $\Phi_e^h(x)\downarrow \neq C(x)$, alors le cas 1 serait vrai en prenant n'importe quel $f \in [h, m_t] \subseteq [g_t, m_t]$. Il s'ensuit que $\Phi_e^h(x)\downarrow = C(x)$. Enfin, remarquons que nous n'avons considéré que des conditions avec le même $m_t$. Comme expliqué plus haut, pour $m_t$ fixé, la relation d'extension est calculable.

Nous avons donc décrit une procédure calculable pour déterminer la valeur de $C(x)$ quel que soit $x$, contredisant l'hypothèse selon laquelle $C$ n'est pas calculable.

Il suffit alors de satisfaire chaque contrat $\mathcal{S}_e$ en étendant petit à petit nos conditions, tout en faisant croître « artificiellement » leurs secondes composantes vers $+\infty$ afin de s'assurer que la solution finale est bien une fonction stable dont la limite est $\emptyset''$. Cela conclut la preuve de la proposition 10.2. ∎

**Corollaire 10.4**
*Il existe un ensemble $A$ à la fois high et Turing-incomplet. Autrement dit, $A' \geqslant_T \emptyset''$ et $A \not\geqslant_T \emptyset'$.*

Preuve. Immédiat par la proposition 10.2, en prenant $C = \emptyset'$. ∎

**Remarque**

L'appellation « condition » est un nom générique emprunté à la théorie du forcing, et qui sera amené à désigner des objets mathématiques très différents tout au long de cet ouvrage, bien que correspondant tous à l'idée d'une approximation d'un objet que l'on construit. Cette notion sera développée dans le chapitre 11.

Notons que l'argument de la preuve précédente ne dépend pas spécifiquement de $\emptyset''$ et l'on aurait pu construire pour tout $B$ un ensemble $A$ tel que $A' \geqslant_T B$ et $A \not\geqslant_T C$. Nous verrons dans le chapitre 13 qu'il existe des ensembles c. e. low non calculables. Sacks [187] a également montré l'existence d'ensembles c. e. incomplets de degrés high.

Il y a une différence de taille entre les ensembles low et high : les premiers sont tous calculables en $\emptyset'$, et ils sont par conséquent en quantité dénombrable. Les deuxièmes peuvent en revanche être arbitrairement complexes, et l'on montre facilement qu'ils sont en quantité indénombrable. Nous verrons en revanche avec le corollaire 19-3.9 et la proposition 10-3.38 qu'il y a « peu » d'ensembles high, du point de vue de la théorie de la mesure et de la théorie des catégories de Baire.

Terminons ce chapitre par quelques exercices.

**Exercice 10.5.** **($\star$)** Adapter la preuve de la proposition 10.2 pour montrer que pour tout ensemble $B$ et tout ensemble non calculable $C$, il existe un ensemble $A$ tel que $A' \geqslant_T B$ et $A \not\geqslant_T C$. ◇

**Exercice 10.6.** **($\star\star$)** Adapter la preuve de la proposition 10.2 pour montrer qu'il existe un ensemble high incomplet et $\emptyset'$-calculable. ◇

**Exercice 10.7.** **($\star\star$)** Montrer par la méthode des extensions finies qu'il existe une suite d'ensembles $(A_n)_{n\in\mathbb{N}}$ deux à deux incomparables pour la réduction Turing, c'est-à-dire tels que

$$A_n \not\leqslant_T A_m \quad \text{et} \quad A_m \not\leqslant_T A_n,$$

pour tous $n \neq m \in \mathbb{N}$. ◇

Chapitre 5

# Hiérarchie arithmétique

On introduit une hiérarchie de complexité sur les ensembles d'entiers. Les ensembles calculatoirement énumérables sont dits $\Sigma^0_1$ et leurs complémentaires sont dits $\Pi^0_1$. Les intersections dénombrables d'ensembles $\Sigma^0_1$ sont dits $\Pi^0_2$ et les réunions dénombrables d'ensembles $\Pi^0_1$ sont dits $\Sigma^0_2$, et ainsi de suite.

On appelle cette construction *hiérarchie arithmétique*, car les ensembles qu'elle contient sont exactement ceux qui sont définissables par une formule du premier ordre dans le langage de l'arithmétique de Peano. Nous reparlerons de cette équivalence dans la section 9-3. Il y a une correspondance directe entre la définition d'un ensemble par réunions/intersections, et le fait de pouvoir le définir par une formule comportant des quantificateurs $\exists/\forall$. Ainsi, une réunion correspond à un quantificateur $\exists$ et une intersection à un quantificateur $\forall$.

**Exemple 1.** L'ensemble des codes $e$ pour les fonctions calculables totales peut s'écrire

$$\{e \in \mathbb{N} \ : \ \forall n \ \exists t \ \Phi_e(n)[t]{\downarrow}\} = \bigcap_n \bigcup_t \{e \in \mathbb{N} \ : \ \Phi_e(n)[t]{\downarrow}\}.$$

Il s'agit d'un ensemble $\Pi^0_2$, c'est-à-dire d'une intersection dénombrable d'ensembles $\Sigma^0_1$, chacun d'entre eux étant par ailleurs une réunion dénombrable d'ensembles calculables.

# 1. Propriétés élémentaires

Commençons par une définition formelle de la hiérarchie arithmétique.

**Définition 1.1.** Soient $m, n \geqslant 1$.

1. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Sigma_n^0$ s'il existe un ensemble calculable $R$ inclus dans $\mathbb{N}^{n+m}$ tel que
$$A = \{(y_1, \dots, y_m) \ : \ \overbrace{\exists x_1 \forall x_2 \dots Q x_n}^{n \text{ quantificateurs}} \ (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$
où $Q$ vaut $\exists$ si $n$ est impair, et $\forall$ si $n$ est pair.
2. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Pi_n^0$ s'il existe un ensemble calculable $R$ inclus dans $\mathbb{N}^{n+m}$ tel que
$$A = \{(y_1, \dots, y_m) \ : \ \overbrace{\forall x_1 \exists x_2 \dots Q x_n}^{n \text{ quantificateurs}} \ (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$
où $Q$ vaut $\forall$ si $n$ est impair, et $\exists$ si $n$ est pair. ♢

Attention dans la définition précédente, c'est l'alternance entre les quantificateurs $\exists$ et $\forall$ qui compte.

Ainsi, par exemple, l'ensemble
$$\{y \ : \ \exists x_1 \forall x_2 \ R(y, x_1, x_2)\}$$
pour $R$ calculable est un ensemble $\Sigma_2^0$, mais l'ensemble
$$\{y \ : \ \exists x_1 \exists x_2 \exists x_3 \ R(y, x_1, x_2, x_3)\}$$
est, malgré ses trois quantificateurs, un ensemble $\Sigma_1^0$ : on peut facilement éliminer les répétitions de quantificateurs du même type. Considérons par exemple une formule de la forme
$$\exists x_1 \ \exists x_2 \ \forall y_1 \ \forall y_2 \ R(x_1, x_2, y_1, y_2).$$
Cette dernière pourra simplement se récrire sous la forme $\exists x \ \forall y \ R'(x, y)$ où $R'$ sera une version de $R$ modifiée, qui considérera $x$ et $y$ respectivement comme des paires $\langle x_1, x_2 \rangle$ et $\langle y_1, y_2 \rangle$. Le prédicat $R'$ utilisera alors les projections $\pi_1$ et $\pi_2$ réalisant les fonctions inverses de la bijection calculable de couplage $\langle \, , \rangle : \mathbb{N}^2 \to \mathbb{N}$, afin de récupérer $x_1, x_2, y_1, y_2$. Le lecteur peut revenir à l'exercice 3-2.3 pour se convaincre que la bijection $\langle \, \rangle$ et que ses deux fonctions inverses sont calculables. Un abus de notation consistera par exemple à écrire $\exists \langle x_1, x_2 \rangle \ \forall \langle y_1, y_2 \rangle \ R(x_1, x_2, y_1, y_2)$, signifiant que le prédicat $R$ se charge de récupérer $x_1, x_2$ à partir de $\langle x_1, x_2 \rangle$, et de même pour $y_1, y_2$.

**Prédicats**

On parlera parfois de prédicats $\Sigma^0_n$ pour désigner une formule de la forme $\exists x_1\ \forall x_2\ \ldots Q x_n\ R(x_1, x_2, \ldots, x_n)$, où $R$ est un prédicat calculable.

Notez qu'en utilisant le fait que le complémentaire d'un ensemble calculable est calculable, on voit facilement que les ensembles $\Sigma^0_n$ sont les complémentaires des ensembles $\Pi^0_n$. Il est par ailleurs tout à fait possible pour un ensemble d'être à la fois $\Sigma^0_n$ et $\Pi^0_n$. On introduit pour cela la notion suivante.

**Définition 1.2.** Soit $m \geqslant 1$. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Delta^0_n$ pour $n > 0$ s'il est à la fois $\Sigma^0_n$ et $\Pi^0_n$. ♢

La hiérarchie arithmétique établit un premier niveau de distinction entre les ensembles arithmétiquement définissables. Intuitivement, un ensemble $\Sigma^0_{n+1}$ est strictement plus complexe qu'un ensemble $\Sigma^0_n$ ou même $\Pi^0_n$. En effet, chacun des deux derniers peut s'écrire sous une forme $\Sigma^0_{n+1}$ en rajoutant simplement des quantificateurs inutilisés.

**Exemple 1.3.** L'ensemble $\Sigma^0_2$ donné par $\{y\ :\ \exists x_1 \forall x_2\ R(y, x_1, x_2)\}$ peut également s'écrire sous une forme $\Pi^0_3$ : $\{y\ :\ \forall z \exists x_1 \forall x_2\ R(y, x_1, x_2)\}$ ou sous une forme $\Sigma^0_3$ : $\{y\ :\ \exists x_1 \forall x_2 \exists z\ R(y, x_1, x_2)\}$.

En revanche, il n'est pas toujours possible d'utiliser moins de quantificateurs. Nous allons montrer avec le corollaire 5.6 qu'il existe pour tout $n > 0$ des ensembles $\Delta^0_{n+1}$ qui ne peuvent être décrits de manière $\Sigma^0_n$ ou $\Pi^0_n$ : la hiérarchie arithmétique est stricte.

Donnons avant de continuer quelques exemples d'ensembles de la hiérarchie arithmétique.

**Exemple 1.5.**
- ▷ Tout ensemble $A \subseteq \mathbb{N}$ calculable est $\Delta^0_1$ (nous en verrons une preuve avec la proposition 3.4). Il suffit de rajouter une quantification existentielle ou universelle inutile au prédicat calculable $A$ pour pouvoir l'exprimer respectivement comme ensemble $\Sigma^0_1$ ou $\Pi^0_1$.
- ▷ Tout ensemble $A \subseteq \mathbb{N}$ calculatoirement énumérable est $\Sigma^0_1$ (nous le verrons précisément avec la proposition 3.3). En effet, un tel $A$ est décrit comme $\{x\ :\ \exists t\ \Phi_e(x)[t]{\downarrow}\}$ pour un certain $e$.
- ▷ Nous avons vu avec l'exemple 1 que l'ensemble des codes de fonctions totales est $\Pi^0_2$. Son complémentaire, à savoir l'ensemble des codes de fonctions partielles, est donc $\Sigma^0_2$ : $\{e\ :\ \exists n\ \forall t\ \Phi_e(n)[t]{\uparrow}\}$

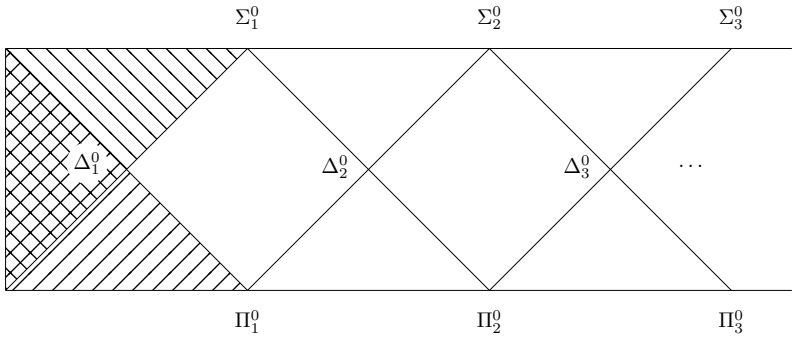



FIGURE 1.4 – *Représentation de la hiérarchie arithmétique. Le triangle supérieur gauche hachuré représente les* $\Sigma_1^0$*. Le triangle inférieur gauche hachuré représente les* $\Pi_1^0$*. L'intersection entre les deux — le triangle doublement hachuré — représente les* $\Delta_1^0$*. Le reste du diagramme continue de la même manière.*

▷ L'ensemble des codes de fonctions qui sont totales, sauf pour un nombre fini d'éléments, est $\Sigma_3^0$ :

$$\{e \ : \ \exists n \ \forall m \ \exists t \ \text{ tel que } m < n \text{ ou } \Phi_e(m)[t]{\downarrow}\}.$$

Nous verrons un peu plus loin que les exemples précédents sont optimaux. Par exemple, l'ensemble des codes de fonctions qui sont totales sauf pour un nombre fini d'éléments ne peut pas s'exprimer de manière $\Pi_3^0$ : il s'agit d'un ensemble $\Sigma_3^0$ strict.

Nous montrons à présent une série de trois propositions sur la stabilité des ensembles $\Sigma_m^0$ et $\Pi_m^0$ par différentes opérations. Par exemple, la stabilité par réunion finie signifie qu'une réunion finie d'ensembles $\Sigma_m^0$ est encore un ensemble $\Sigma_m^0$. Les propositions seront démontrées uniquement en considérant des sous-ensembles de $\mathbb{N}$, mais se généralisent sans problème aux sous-ensembles de $\mathbb{N}^n$ pour $n$ arbitraire.

**Proposition 1.6.** Les ensembles $\Sigma_m^0$ (resp. $\Pi_m^0$) sont stables par réunions finies et intersections finies. ⋆

PREUVE. Soit $m > 0$. Soient

$$\begin{aligned} A_0 &= \{n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R_0(n, x_1, \ldots, x_m)\} \\ A_1 &= \{n \ : \ \exists y_1 \ \forall y_2 \ \ldots \ Q y_m \ R_1(n, y_1, \ldots, y_m)\} \end{aligned}$$

deux ensembles $\Sigma_m^0$, où $Q$ est le symbole $\exists$ si $m$ est impair et $\forall$ si $m$ est pair. On laisse au lecteur le soin de montrer, par inclusion dans un sens

puis dans l'autre, que :

$$A_0 \cap A_1 = \Big\{ n : \begin{array}{l} \exists \langle x_1, y_1 \rangle \ \forall \langle x_2, y_2 \rangle \ \ldots \ Q \langle x_m, y_m \rangle \\ \text{tels que } R_0(n, x_1, \ldots, x_m) \wedge R_1(n, y_1, \ldots, y_m) \end{array} \Big\}.$$

Les ensembles $\Sigma^0_m$ sont donc stables par intersections finies. On a par ailleurs de la même manière :

$$A_0 \cup A_1 = \Big\{ n : \begin{array}{l} \exists \langle x_1, y_1 \rangle \ \forall \langle x_2, y_2 \rangle \ \ldots \ Q \langle x_m, y_m \rangle \\ \text{tels que } R_0(n, x_1, \ldots, x_m) \vee R_1(n, y_1, \ldots, y_m) \end{array} \Big\}.$$

Les ensembles $\Sigma^0_m$ sont donc stables par réunions finies. Par passage au complémentaire, les ensembles $\Pi^0_m$ sont eux aussi stables par réunions et intersections finies. ∎

Souvenons-nous du concept de suite uniformément calculable, introduit à l'occasion des développements qui suivent le lemme 4-7.2. Le concept d'uniformité passe à la hiérarchie arithmétique de la manière suivante : dans la prochaine proposition, une réunion dénombrable $(A_i)_{i \in \mathbb{N}}$ d'ensembles *uniformément* $\Sigma^0_m$ est donc une réunion telle que chaque ensemble $A_i$ admet la même description $\Sigma^0_m$, mais avec comme paramètre $i$. Formellement, si

$$A_i = \{ n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R_i(n, x_1, \ldots, x_m) \},$$

il faut qu'il existe un processus calculable permettant à partir de $i$ de renvoyer un code pour le prédicat $R_i$. De manière équivalente, on peut considérer que $A_i$ est décrit comme

$$A_i = \{ n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R(i, n, x_1, \ldots, x_m) \},$$

où $R$ est un prédicat calculable prenant $i$ comme paramètre supplémentaire.

**Proposition 1.7.** Les ensembles $\Sigma^0_m$ (resp. $\Pi^0_m$) sont stables par réunions dénombrables uniformes (resp. intersections dénombrables uniformes). ⋆

Preuve. Soit $m > 0$, et soit $(A_i)_{i \in \mathbb{N}}$ une suite d'ensembles uniformément $\Sigma^0_m$ :

$$A_i = \{ n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R(i, n, x_1, \ldots, x_m) \},$$

où $Q$ est le symbole $\exists$ si $m$ est impair et $\forall$ si $m$ est pair. On montre aisément par inclusion dans un sens puis dans l'autre que

$$\bigcup_i A_i = \{ n \ : \ \exists \langle i, x_1 \rangle \ \forall x_2 \ \ldots \ Q x_m \ R(i, n, x_1, \ldots, x_m).$$

Les ensembles $\Sigma^0_m$ sont donc stables par réunions dénombrables uniformes. Par passage au complémentaire, les ensembles $\Pi^0_m$ sont eux aussi stables par intersections dénombrables uniformes. ∎

Notons que la stabilité des ensembles $\Sigma^0_n$ par réunion dénombrable uniforme équivaut à la stabilité par quantification existentielle (resp. quantification universelle pour les ensembles $\Pi^0_n$).

**Exercice 1.8.** Montrer que les ensembles $\Sigma^0_n$ ne sont pas stables par réunions dénombrables non uniformes. ◇

On montre à présent la stabilité par quantification bornée uniforme. Une quantification bornée de la forme $\forall x < m$ peut être vue comme une réunion finie de $m$ ensembles paramétrés par $x$. La proposition suivante n'est toutefois pas identique à la proposition 1.6 : on veut ici l'uniformité en fonction de la borne qui peut elle-même être une variable, ou dépendre d'autres variables sur lesquelles on quantifie. Grosso modo, ce que dit la proposition suivante est que l'ensemble

$$\{n \ : \ \exists x \ \forall y < x \ \exists t \ R(n,x,y,t)\},$$

où $R$ est un prédicat calculable, peut se récrire comme

$$\{n \ : \ \exists x \ \exists t \ \forall y < x \ \exists s < t \ R(n,x,y,s)\}.$$

Le prédicat $\forall y < x \ \exists s < t \ R(n,x,y,s)$ étant calculable, l'ensemble est $\Sigma^0_1$.

**Proposition 1.9.** Les ensembles $\Sigma^0_m$ et $\Pi^0_m$ sont stables par quantifications bornées uniformes. ⋆

Preuve. On montre que l'on peut toujours déplacer la quantification bornée vers la droite, jusqu'à ce que celle-ci se retrouve à côté du prédicat calculable.

Soit $A = \{(n,k) \ : \ \forall y < k \ \exists x \ R(n,y,x)\}$, où $R$ est un ensemble calculable ou $\Pi^0_m$ pour $m > 0$. Alors, on a aussi

$$A = \{(n,k) \ : \ \exists x \ \forall y < k \ \exists z < x \ R(n,y,z)\}.$$

L'égalité vient du fait que si pour tout $y < k$ un certain $x_k$ témoigne qu'une formule est vraie, comme le nombre de témoins est fini, ceux-ci sont bornés dans $\mathbb{N}$. La variable $x$ qui faisait auparavant office de témoin est à présent utilisée comme borne sur tous les témoins possibles. On a de la même manière par passage au complémentaire

$$\{(n,k) \ : \ \exists y < k \ \forall x \ R(n,y,x)\} = \{(n,k) \ : \ \forall x \ \exists y < k \ \forall z < x \ R(n,y,z)\},$$

où $R$ est un ensemble calculable ou $\Sigma^0_m$.

Si l'on a à présent $A = \{(n,k) \ : \ \exists y < k \ \exists x \ R(n,y,x)\}$, où $R$ est un ensemble calculable ou $\Pi^0_m$, alors on a aussi $A = \{(n,k) \ : \ \exists x \ \exists y < k R(n,y,x)\}$ de manière immédiate. On a de la même manière par passage au complémentaire $\{(n,k) \ : \ \forall y < k \ \forall x \ R(n,y,x)\} = \{(n,k) \ : \ \forall x \ \forall y < k \ R(n,y,x)\}$, où $R$ est un ensemble calculable ou $\Sigma^0_m$.

Par récurrence, on déplace ainsi les quantifications bornées vers la droite jusqu'à ce que celles-ci soient toutes collées au prédicat calculable. On utilise enfin le fait que les prédicats calculables sont stables par quantifications bornées (voir l'exercice 3-2.4) pour conclure. ∎

## 2. Hiérarchie arithmétique et calculabilité

Voyons à présent à quoi correspondent les premiers niveaux de la hiérarchie arithmétique.

**Proposition 2.1.** Un ensemble $A \subseteq \mathbb{N}$ est $\Sigma^0_1$ ssi il est calculatoirement énumérable. ⋆

Preuve. Supposons que l'ensemble $A$ soit calculatoirement énumérable. Alors, $A = \{n \ : \ \exists t \ \Phi_e(n)[t]\downarrow\}$ pour un certain $e$. Le prédicat $\Phi_e(n)[t]\downarrow$ est calculable. Donc, $A$ est $\Sigma^0_1$. Supposons à présent que $A$ soit $\Sigma^0_1$. Soit $A = \{n \ : \ \exists t \ R(n,t)\}$, où $R$ est un prédicat calculable. Alors, on définit facilement la machine de code $e$ qui sur l'entrée $n$ cherche le plus petit $t$ tel que $R(n,t)$ et s'arrête, ou continue sa recherche indéfiniment sinon. On a alors $\Phi_e(n)\downarrow$ ssi $\exists t \ R(n,t)$. ∎

**Proposition 2.2.** Un ensemble $A \subseteq \mathbb{N}$ est $\Delta^0_1$ ssi il est calculable. ⋆

Preuve. Un ensemble $A$ est $\Delta^0_1$ ssi $A$ et $\mathbb{N} \setminus A$ sont $\Sigma^0_1$ ssi $A$ et $\mathbb{N} \setminus A$ sont calculatoirement énumérables (d'après la proposition 2.1), ou encore ssi $A$ est calculable (d'après la proposition 3-7.4). ∎

Nous avons vu qu'il y a des ensembles calculatoirement énumérables qui ne sont pas calculables, et en particulier dont le complémentaire n'est pas calculatoirement énumérable. Cela implique que certains ensembles sont $\Sigma^0_1$ mais pas $\Delta^0_1$, et en particulier pas $\Pi^0_1$. Nous allons voir dans les prochaines sections que la hiérarchie est stricte partout : pour tout $n$, il existe des ensembles $\Sigma^0_n$ qui ne sont pas $\Pi^0_n$, et il existe des ensembles $\Delta^0_{n+1}$ qui ne sont ni $\Sigma^0_n$, ni $\Pi^0_n$.

Intuitivement, le nombre de quantificateurs correspond au « nombre de fois qu'il faudrait compter jusqu'à l'infini » pour déterminer l'appartenance d'un élément à l'ensemble. Ainsi, pour un ensemble $\Sigma^0_1$ s'écrivant comme $\{n \ : \ \exists t \ R(t,n)\}$, où $R$ est un prédicat calculable, il faudrait tester la valeur de vérité de $R(t,n)$ pour tous les entiers $t$, afin d'en trouver un qui témoigne de l'appartenance de $n$ à l'ensemble, ou bien afin d'être sûr que $n$ n'y appartient pas.

Pour un ensemble $\Pi^0_2$ de la forme

$$\{n \ : \ \forall t_1 \ \exists t_2 \ R(t_1, t_2, n)\},$$

il faudrait une première procédure qui teste tous les entiers $t_1$, et qui pour chacun de ces tests, examine la valeur de vérité de $R(t_1, t_2, n)$ pour tous les entiers $t_2$. Cela correspondrait en quelque sorte à deux boucles imbriquées se déroulant chacune sur l'ensemble de tous les entiers.

## 3. Relativisation à un oracle

La hiérarchie arithmétique permet de définir des ensembles de plus en plus complexes, et dans un sens de moins en moins calculables. Toutefois, la classe des ensembles arithmétiquement définissables reste dénombrable. Il reste donc « beaucoup » d'ensembles, pour ainsi dire la majorité, qui ne peuvent être ni calculés, ni même définis par une formule de l'arithmétique. Il est évidemment difficile d'en parler, et toute tentative de les cerner plus précisément les rendraient définissables dans un certain langage, élargissant simplement un peu plus la classe immanquablement dénombrable des ensembles dont on arrive à dire quelque chose, laissant de côté la majorité des autres ensembles, cachés, inaccessibles.

Nous contournerons le problème tout au long des chapitres à venir, essentiellement en étudiant des « groupes d'ensembles » plutôt que chaque ensemble individuellement. Nous verrons en particulier dans les chapitres à venir de nombreuses propriétés calculatoires partagées par certains ensembles, en général une quantité indénombrable d'entre eux. Nous mènerons ensuite dans la partie II une étude des ensembles typiques du point de vue de la théorie de la mesure, c'est-à-dire des ensembles que l'on obtient avec probabilité 1 si l'on sélectionne leurs bits au hasard.

Pour le moment, nous nous contentons de relativiser la hiérarchie arithmétique à un oracle : étant donné un ensemble $X$ quelconque, on considère les ensembles $\Sigma^0_n, \Pi^0_n$ et $\Delta^0_n$ que l'on peut définir relativement à la connaissance de $X$. On se base pour cela sur les calculs avec oracle du théorème 4-2.2, et l'on itère comme dans la définition 1.1.

**Définition 3.1.** Soient $m, n \geqslant 1$. Soit $X \subseteq \mathbb{N}$.

1. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Sigma^0_n(X)$ s'il existe un ensemble $X$-calculable $R \subseteq \mathbb{N}^{n+m}$ tel que

$$A = \{(y_1, \dots, y_m) \ : \ \overbrace{\exists x_1 \forall x_2 \dots Q x_n}^{n \text{ quantificateurs}} \ (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

où $Q$ vaut $\exists$ si $n$ est impair, et $\forall$ si $n$ est pair.

2. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Pi^0_n(X)$ s'il existe un ensemble $X$-calculable $R \subseteq \mathbb{N}^{n+m}$ tel que

$$A = \{(y_1, \dots, y_m) \ : \ \overbrace{\forall x_1 \exists x_2 \dots Q x_n}^{n \text{ quantificateurs}} \ (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

où $Q$ vaut $\forall$ si $n$ est impair, et $\exists$ si $n$ est pair. ♢

**Définition 3.2.** Soit $m \geqslant 1$. Soit $X \subseteq \mathbb{N}$. Un ensemble $A \subseteq \mathbb{N}^m$ est dit $\Delta^0_n(X)$ pour $n > 0$ s'il est à la fois $\Sigma^0_n(X)$ et $\Pi^0_n(X)$. ♢

Les propositions suivantes se montrent comme leurs équivalents respectifs de la section précédente.

**Proposition 3.3.** Un ensemble $A \subseteq \mathbb{N}$ est $\Sigma^0_1(X)$ ssi il est $X$-c. e. ⋆

**Proposition 3.4.** Un ensemble $A \subseteq \mathbb{N}$ est $\Delta^0_1(X)$ ssi il est $X$-calculable. ⋆

Notons qu'un oracle pourra fournir de la puissance de calcul supplémentaire, permettant à certains ensembles normalement non $\Sigma^0_n$, de devenir $\Sigma^0_n(X)$. On peut même par exemple définir un oracle $X$ tel que tous les ensembles arithmétiques, c'est-à-dire $\Sigma^0_n$ pour un certain $n$, deviennent $\Delta^0_1(X)$. Toutefois, quelle que soit la puissance de calcul de $X$, les ensembles arithmétiques en $X$ restent en quantité dénombrable.

# 4. Degrés many-one

La classification de la hiérarchie arithmétique en termes d'ensembles $\Sigma^0_n$ et $\Pi^0_n$ n'est pas une notion de degrés Turing, car un ensemble $\Sigma^0_n$ peut être Turing équivalent à un ensemble qui ne l'est pas. De fait, tout ensemble $\Sigma^0_n$ $A$ est Turing équivalent à son complémentaire $\Pi^0_n$ $\overline{A}$. En ce sens, la réduction Turing est « grossière », car elle ne distingue pas un ensemble de son complémentaire.

Nous allons introduire une notion plus fine que la réduction Turing, au sens où elle implique cette dernière. Il s'agit de la réduction many-one, qui comme nous le verrons préserve la hiérarchie arithmétique. Dans les faits, beaucoup de preuves de réduction Turing que nous avons vues sont des réductions many-one.

**Définition 4.1.** Soient deux ensembles $A, B$. On dit que $A$ est *many-one réductible* à $B$, et l'on écrit $A \leqslant_m B$ s'il existe une fonction totale calculable $f : \mathbb{N} \to \mathbb{N}$ telle que $n \in A \leftrightarrow f(n) \in B$. Si $A \leqslant_m B$ et $B \leqslant_m A$, on écrit $A \equiv_m B$. On écrit $A <_m B$ si $A \leqslant_m B$ mais $B \not\leqslant_m A$. On appelle *degrés many-one* les classes d'équivalence de la relation $\equiv_m$. ♢

Quand on a $A \leqslant_m B$, la connaissance de $B$ est suffisante pour calculer $A$, et l'on a en particulier $A \leqslant_T B$ : soit $f$ la fonction totale calculable telle que $n \in A \leftrightarrow f(n) \in B$. Alors, pour savoir si $n \in A$, on utilise $f$ pour « poser une question » à l'ensemble $B$ : est-ce que $f(n) \in B$ ? Si la réponse est oui, alors $n \in A$. Sinon, $n \notin A$. La réduction many-one est très restrictive : si $A \leqslant_m B$, alors pour connaître un bit de $A$ on n'a le droit de poser qu'une seule question à l'oracle $B$. Comme si cela ne suffisait pas, la réponse à la question détermine directement l'appartenance du bit à $A$ sans qu'il ne soit possible d'inverser cette décision. L'importance de cette réduction, très restrictive, vient du fait qu'elle préserve la hiérarchie arithmétique.

**Proposition 4.2.** Soit $A \subseteq \mathbb{N}$ un ensemble $\Sigma^0_m(X)$ (resp. $\Pi^0_m(X)$) pour un certain $X \subseteq \mathbb{N}$, et soit $B \leqslant_m A$. Alors, $B$ est $\Sigma^0_m(X)$ (resp. $\Pi^0_m(X)$). ⋆

Preuve. Soit $A$ un ensemble $\Sigma^0_m(X)$. Alors,

$$A = \{n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R(n, x_1, \ldots, x_m)\},$$

où $R$ est un ensemble $X$-calculable. Soit $f$ la fonction totale calculable telle que $n \in B$ ssi $f(n) \in A$. On a donc

$$B = \{n \ : \ \exists x_1 \ \forall x_2 \ \ldots \ Q x_m \ R(f(n), x_1, \ldots, x_m)\},$$

ce qui est bien une description $\Sigma^0_m(X)$ de $B$.

On montre de la même manière que si $A$ est $\Pi^0_m(X)$ et $B \leqslant_m A$, alors $B$ est $\Pi^0_m(X)$. ∎

Parmi les ensembles $\Sigma^0_1$, l'arrêt des programmes informatiques joue un rôle particulier : il est le plus « puissant » des ensembles $\Sigma^0_1$, dans le sens où tout ensemble $\Sigma^0_1$ est many-one réductible à l'arrêt.

**Proposition 4.3.** Un ensemble $A$ est $\Sigma^0_1$ si, et seulement si, $A \leqslant_m \emptyset'$. ⋆

Preuve. Supposons $A$ $\Sigma^0_1$. Alors, il existe $e$ tel que $n \in A$ ssi $\Phi_e(n)\downarrow$. En utilisant le théorème SMN, on définit une fonction calculable $s : \mathbb{N} \to \mathbb{N}$ telle que pour tout $m$ on ait $\Phi_e(n) = \Phi_{s(n)}(m)$. On aura en particulier $n \in A$ ssi $\Phi_e(n)\downarrow$, ou ce qui revient au même ssi $\Phi_{s(n)}(s(n))\downarrow$, ou encore ssi $s(n) \in \emptyset'$. Donc, $A \leqslant_m \emptyset'$.

Supposons à présent $A \leqslant_m \emptyset'$. Alors, comme $\emptyset'$ est $\Sigma^0_1$, l'ensemble $A$ est lui aussi $\Sigma^0_1$, d'après la proposition 4.2. ∎

On dit aussi que $\emptyset'$ est un ensemble $\Sigma^0_1$-complet, comme nous le verrons avec la définition 5.2.

# 5. Théorème de Post

Nous allons voir qu'il y a une correspondance précise entre les itérations du saut Turing et la hiérarchie arithmétique.

**Définition 5.1.** Étant donné un ensemble $X \subseteq \mathbb{N}$, on définit récursivement sur $n \geqslant 0$ :

1. $X^{(0)} = X$
2. $X^{(n+1)} = X^{(n)'}$. ♢

Ainsi, $\emptyset^{(1)} = \emptyset'$, $\emptyset^{(2)} = \emptyset''$, etc. Nous allons à présent montrer que, pour tout $n$, l'ensemble $\emptyset^{(n)}$ est $\Sigma^0_n$-complet : il s'agit d'un ensemble $\Sigma^0_n$, qui a également une puissance calculatoire maximale parmi les ensembles $\Sigma^0_n$, dans le sens où tout ensemble $\Sigma^0_n$ est many-one réductible à $\emptyset^{(n)}$.

**Définition 5.2.** Un ensemble $A \subseteq \mathbb{N}$ est $\Sigma^0_n(X)$-*complet* (resp. $\Pi^0_n(X)$-*complet*) s'il est $\Sigma^0_n(X)$ (resp. $\Pi^0_n(X)$) et si, pour tout ensemble $\Sigma^0_n(X)$ (resp. $\Pi^0_n(X)$) $B$, on a $B \leqslant_m A$. ♢

**Proposition 5.3.** Soit $n > 0$. L'ensemble $\emptyset^{(n)}$ est $\Sigma^0_n$-complet. De la même manière, pour tout ensemble $X \subseteq \mathbb{N}$, l'ensemble $X^{(n)}$ est $\Sigma^0_n(X)$-complet.⋆

Preuve. Montrons par récurrence sur $n$ que pour tout $n > 0$ l'ensemble $\emptyset^{(n)}$ est $\Sigma^0_n$. Par définition, l'ensemble $\emptyset^{(1)}$ est $\Sigma^0_1$. Supposons que l'ensemble $\emptyset^{(n)}$ soit $\Sigma^0_n$. Alors, l'ensemble $\emptyset^{(n+1)}$ est défini comme :

$$\left\{ m \ : \ \exists t \in \mathbb{N} \ \exists \sigma \in 2^{<\mathbb{N}} \quad \begin{array}{ll} \Big( \forall s < |\sigma| & \Big( \sigma(s) = 1 \text{ et } s \in \emptyset^{(n)} \Big) \\ \text{ou} & \Big( \sigma(s) = 0 \text{ et } s \notin \emptyset^{(n)} \Big) \Big) \\ \text{et} \quad \Phi_m(\sigma, m)[t]\downarrow & \end{array} \right\}.$$

La description de $\emptyset^{(n+1)}$ se fait donc avec des quantificateurs existentiels, suivi d'un prédicat utilisant $s \in \emptyset^{(n)}$, ce qui est par récurrence $\Sigma^0_n$, et utilisant $s \notin \emptyset^{(n)}$, ce qui est par récurrence $\Pi^0_n$, et donc $\Sigma^0_{n+1}$. En utilisant les propriétés de stabilité des propositions 1.9 et 1.6, le prédicat qui suit les quantificateurs existentiels $\exists t \in \mathbb{N} \ \exists \sigma \in 2^{<\mathbb{N}}$ est en particulier $\Sigma^0_{n+1}$ uniformément en $m, \sigma$ et $t$. En utilisant à présent la stabilité par réunion dénombrable uniforme de la proposition 1.7, l'ensemble $\emptyset^{(n+1)}$ est donc $\Sigma^0_{n+1}$.

Montrons à présent par récurrence sur $n$ que tout ensemble $\Sigma^0_n$ est many-one réductible à $\emptyset^{(n)}$.
C'est le cas par la proposition 4.3 pour $n = 1$. Supposons que ce soit le cas pour un certain $n$. Soit $A = \{x \ : \ \exists y \ R(x, y)\}$, où $R$ est un ensemble $\Pi^0_n$.

Par récurrence, il existe une fonction totale calculable $g : \mathbb{N} \to \mathbb{N}$ telle que $(x, y) \in R$ ssi $g(\langle x, y\rangle) \notin \emptyset^{(n)}$. On définit la fonction totale calculable $f$ telle que $\Phi_{f(x)}^{\emptyset^{(n)}}$ s'arrête sur toute entrée si $\exists y\ g(\langle x, y\rangle) \notin \emptyset^{(n)}$, et ne s'arrête sur aucune entrée sinon. On a donc $f(x) \in \emptyset^{(n+1)}$ ssi $\Phi_{f(x)}(\emptyset^{(n)}, f(x))\downarrow$, autrement dit ssi $\exists y\ g(\langle x, y\rangle) \notin \emptyset^{(n)}$, ou encore ssi $\exists y\ R(x, y)$, ou enfin ssi $x \in A$. L'ensemble $A$ est donc many-one réductible à $\emptyset^{(n+1)}$.

La relativisation à un oracle $X$ est similaire et ne présente pas de difficulté particulière. ∎

**Corollaire 5.4**
*Un ensemble $A$ est $\Sigma_n^0(X)$ ssi $A \leqslant_m X^{(n)}$.*

Preuve. Par la proposition précédente et par la définition de la $\Sigma_n^0(X)$-complétude. ∎

Nous arrivons finalement au théorème de Post.

**Théorème 5.5 (Théorème de Post)**
*Soient $A$ un ensemble et $n \geqslant 0$.*

*(1) $A$ est $\Sigma_{n+1}^0$ ssi $A$ est $\Sigma_1^0(\emptyset^{(n)})$, ou encore ssi $A$ est $\emptyset^{(n)}$-c. e.*

*(2) $A$ est $\Delta_{n+1}^0$ ssi $A$ est $\Delta_1^0(\emptyset^{(n)})$, ou encore ssi $A \leqslant_T \emptyset^{(n)}$*

Preuve. Montrons (1). Par le corollaire 5.4, $A$ est $\Sigma_{n+1}^0$ ssi $A \leqslant_m \emptyset^{(n+1)}$. En relativisant la proposition 4.3 à $\emptyset^{(n)}$, on obtient $A \leqslant_m \emptyset^{(n)'}$ (qui est égal à $\emptyset^{(n+1)}$) ssi $A$ est $\Sigma_1^0(\emptyset^{(n)})$. Enfin, d'après la proposition 3.3, $A$ est $\Sigma_1^0(\emptyset^{(n)})$ ssi $A$ est $\emptyset^{(n)}$-c. e.

Montrons (2). Par définition, $A$ est $\Delta_{n+1}^0$ ssi $A$ et $\overline{A}$ sont tous les deux $\Sigma_{n+1}^0$. Par le point précédent, c'est équivalent à dire que $A$ et $\overline{A}$ sont $\emptyset^{(n)}$-c. e. En relativisant la proposition 3-7.4 à $\emptyset^{(n)}$, $A$ et $\overline{A}$ sont $\emptyset^{(n)}$-c. e. ssi $A \leqslant_T \emptyset^{(n)}$. Enfin, d'après la proposition 3.4, on a $A \leqslant_T \emptyset^{(n)}$ ssi $A$ est $\Delta_1^0(\emptyset^{(n)})$. ∎

**Corollaire 5.6**
*La hiérarchie arithmétique est stricte. En d'autres termes,*

- ▷ *pour tout $n > 0$, il existe un ensemble $\Sigma_n^0$ qui n'est pas $\Pi_n^0$ et un ensemble $\Pi_n^0$ qui n'est pas $\Sigma_n^0$ ;*
- ▷ *pour tout $n > 0$, il existe un ensemble $\Delta_{n+1}^0$ qui n'est ni $\Sigma_n^0$ ni $\Pi_n^0$.*

Preuve. D'après la proposition 5.3, l'ensemble $\emptyset^{(n)}$ est $\Sigma^0_n$. Il ne peut pas être $\Pi^0_n$ auquel cas il serait $\Delta^0_n$, et donc calculable en $\emptyset^{(n-1)}$ d'après le théorème 5.5, contredisant le fait que $X$ ne calcule jamais son saut Turing. On montre de la même manière que $\mathbb{N} \setminus \emptyset^{(n)}$ est $\Pi^0_n$, mais pas $\Sigma^0_n$.

On peut enfin construire pour tout $n$ l'ensemble $\Delta^0_{n+1}$ suivant : l'ensemble $X$ tel que $X(2m) = \emptyset^{(n)}(m)$ et tel que $X(2m+1) = (\mathbb{N} \setminus \emptyset^{(n)})(m)$. Il est clair que $X$ est $\emptyset^{(n)}$-calculable, et donc $\Delta^0_{n+1}$ d'après le théorème 5.5. Enfin, si l'on suppose par l'absurde que $X$ est $\Sigma^0_n$ (resp. $\Pi^0_n$), cela permet de donner une description $\Sigma^0_n$ de $\mathbb{N} \setminus \emptyset^{(n)}$ en ne gardant que les bits impairs de $X$ (resp. une description $\Pi^0_n$ de $\emptyset^{(n)}$ en ne gardant que les bits pairs de $X$), ce qui est en contradiction avec le fait que $\emptyset^{(n)}$ n'est pas $\Pi^0_n$ (resp. avec le fait que $\mathbb{N} \setminus \emptyset^{(n)}$ n'est pas $\Sigma^0_n$). ∎

**Exercice 5.7. (⋆)** Montrer que, pour tous $X, Y \in 2^{\mathbb{N}}$, on a $X \equiv_T Y$ ssi $X' \equiv_m Y'$. ◇

## 6. Théorème de Rice

Le théorème de Rice dit en substance qu'aucune propriété sémantique des programmes n'est décidable. Par exemple, comme vu dans l'exercice 3-6.4, il est impossible de décider si un programme informatique effectue une multiplication par 2 ou pas. Nous entendons *propriétés sémantiques* par opposition à *propriétés syntaxiques*. Ces dernières sont sensibles aux variations de code, par exemple « ce programme possède trois variables distinctes » ou « ce programme contient deux boucles `for` ». Les propriétés sémantiques ne parlent pas directement des programmes, mais des fonctions qu'ils représentent. Par exemple, les propriétés « cete fonction est définie sur l'entrée 42 » ou « cette fonction ne renvoie que des valeurs paires » sont des propriétés sémantiques. Elles ne dépendent pas des détails d'implémentation des fonctions concernées.

**Définition 6.1.** Un *ensemble sémantique de codes* est un ensemble $A$ inclus dans $\mathbb{N}$, tel que pour tous $x, y \in \mathbb{N}$,

$$\text{si} \quad x \in A \quad \text{et} \quad \Phi_x = \Phi_y, \quad \text{alors} \quad y \in A.$$ ◇

Parmi les ensembles sémantiques de codes, on notera l'ensemble vide $\emptyset$ qui correspond à une propriété jamais satisfaite, et l'ensemble $\mathbb{N}$ représentant une propriété toujours vraie. Un ensemble sémantique de codes est *non trivial* si $A \neq \emptyset$ et $A \neq \mathbb{N}$.

**Théorème 6.2**
*Si $A$ est un ensemble sémantique de codes non trivial, alors soit $\emptyset' \leqslant_m A$ soit $\emptyset' \leqslant_m \overline{A}$.*

Preuve. Soit $\Phi_{e_0}$ la fonction nulle part définie. Supposons que $e_0 \in \overline{A}$, l'autre cas étant traité par symétrie. Comme $A$ est non trivial, $A$ est non vide. Fixons un code $e_1 \in A$. En particulier, $\Phi_{e_0} \neq \Phi_{e_1}$. Par le théorème SMN (voir le théorème 3-4.1), il existe une fonction $f : \mathbb{N} \to \mathbb{N}$ totale et calculable telle que

$$\Phi_{f(x)}(y) = \begin{cases} \Phi_{e_1}(y) & \text{si } x \in \emptyset' \\ \uparrow & \text{si } x \notin \emptyset'. \end{cases}$$

Montrons que la fonction $f$ est une réduction many-one de $\emptyset'$ à $A$. Si $x \in \emptyset'$, alors $\Phi_{f(x)} = \Phi_{e_1}$, et ainsi $f(x) \in A$. Inversement, si $x \notin \emptyset'$, $\Phi_{f(x)} = \Phi_{e_0}$, donc $f(x) \in \overline{A}$. ∎

Le théorème de Rice affirme qu'aucune propriété non triviale sur les fonctions partielles calculables n'est décidable. Bien que les applications de ce théorème soient relativement limitées en calculabilité, le théorème de Rice revêt une grande importance pour la compréhension qu'il apporte sur la nature de la calculabilité. En particulier, l'indécidabilité du problème de l'arrêt n'est pas un phénomène isolé, car il est partagé par toutes les propriétés sur les fonctions partielles calculables.

**Corollaire 6.3 (Théorème de Rice)**
*Soit $\mathcal{C}$ une classe de fonctions partielles $\mathbb{N} \to \mathbb{N}$ calculables. Alors, l'ensemble $A = \{x : \Phi_x \in \mathcal{C}\}$ est incalculable sauf si $\mathcal{C} = \emptyset$ ou $\mathcal{C}$ est la classe de toutes les fonctions partielles calculables.*

Preuve. L'ensemble $A$ est un ensemble sémantique de codes. Si $\mathcal{C} = \emptyset$ ou $\mathcal{C}$ est la classe de toutes les fonctions partielles calculables, alors $A = \emptyset$ ou $A = \mathbb{N}$ et, dans les deux cas, $A$ est calculable. Si $\mathcal{C}$ n'est ni vide ni la classe de toutes les fonctions partielles calculables, alors $A$ est non trivial, et par le théorème 6.2 soit $\emptyset' \leqslant_m A$, soit $\emptyset' \leqslant_m \overline{A}$. Dans les deux cas, $\emptyset' \leqslant_T A$, et $A$ est non calculable. ∎

# 7. Codes arithmétiques

Les ensembles d'entiers sont de manière générale des objets infinis, et ne peuvent donc pas être représentés par des entiers naturels sans que certains d'entre eux ne soient omis de cette représentation. C'est l'objet de l'argument diagonal de Cantor (voir la section 2-4). Certains ensembles d'entiers peuvent cependant être décrits de manière finitaire, à commencer par les ensembles calculables.

**Définition 7.1.** Un *code* $\Delta^0_1$ d'un ensemble calculable $A$ est un entier $e$ tel que $\Phi_e = A$ (c'est-à-dire $\forall n\ \Phi_e(n)\downarrow = A(n)$). ♢

Notons qu'un ensemble est calculable ssi il a un code $\Delta^0_1$. Par le lemme 3 -5.1 de remplissage, tout ensemble calculable est représenté par une infinité de codes $\Delta^0_1$. D'après le théorème de Rice, l'ensemble des codes $\Delta^0_1$ d'un ensemble calculable fixé n'est pas décidable. Il n'existe même pas de procédure pour décider si un entier $e$ est un code $\Delta^0_1$ d'un ensemble. En effet, cela reviendrait à être capable d'énumérer de manière calculable tous les ensembles calculables, et nous ferait tomber sous la coupe de l'argument diagonal de Cantor.

**Exercice 7.2. (⋆)** Soit TOT l'ensemble des codes $\Delta^0_1$ ; autrement dit,

$$\text{TOT} = \{e : \Phi_e \text{ est total}\}$$

(en supposant sans perte de généralité que $\Phi_e(n)$ renvoie 0 ou 1 pour tout $n$). Prouver que TOT est $\Pi^0_2$-complet, c.-à-d. que $\mathbb{N} \setminus \emptyset'' \leqslant_m \text{TOT}$. ⋄

Les codes $\Delta^0_1$ permettent de donner une nouvelle définition d'une suite d'ensembles uniformément calculable. Rappelons qu'une suite $X_0, X_1, \ldots$ d'ensembles est uniformément calculable si la fonction $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ définie par $f(x, s) = 1$ ssi $x \in X_s$ est totale calculable.

**Proposition 7.3.** Une suite $X_0, X_1, \ldots$ d'ensembles est uniformément calculable si, et seulement si, il existe une suite calculable $e_0, e_1, \ldots$ tel que $e_s$ est un code $\Delta^0_1$ de $X_s$ pour tout $s$. ⋆

Preuve. $\Rightarrow$. Supposons que $X_0, X_1, \ldots$ soit uniformément calculable à travers la fonction $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$. Par le théorème SMN, il existe une fonction totale calculable $h : \mathbb{N} \to \mathbb{N}$ telle que pour tous $e, x$, on ait l'égalité $\Phi_{h(e)}(x) = f(x, e)$. Il s'ensuit que la suite $h(0), h(1), \ldots$ est une suite calculable telle que $h(s)$ est un code $\Delta^0_1$ de $X_s$.

$\Leftarrow$. Supposons maintenant qu'il existe une suite calculable $e_0, e_1, \ldots$ de codes $\Delta^0_1$. Alors, vu l'existence d'une machine universelle, la fonction $f$ définie sur $\mathbb{N} \times \mathbb{N}$ par $f(x, s) = \Phi_{e_s}(x) \in \mathbb{N}$ est totale calculable, et montre que la suite d'ensembles $X_0, X_1, \ldots$ est uniformément calculable. ∎

Les ensembles calculatoirement énumérables sont également représentables par un système de code. On utilisera pour cela souvent la notation ci-après.

**Notation**

On note $W_e$ l'ensemble $\{n \in \mathbb{N} \ : \ \Phi_e(n)\downarrow\}$, lequel est calculatoirement énumérable. La notation se relativise à un oracle $X$, et $W_e^X$ ou $W_e(X)$ dénotera ainsi l'ensemble $\{n \in \mathbb{N} \ : \ \Phi_e(X, n)\downarrow\}$.

**Définition 7.4.** Un *code* $\Sigma^0_1$ d'un ensemble c. e. $A$ est un entier $e$ tel que $W_e = A$. ♢

Toujours par le théorème de Rice, l'ensemble des codes $\Sigma^0_1$ d'un ensemble c. e. fixé n'est pas calculable. Cependant, contrairement aux codes $\Delta^0_1$, *tout entier* est un code $\Sigma^0_1$. Les ensembles calculables étant *a fortiori* calculatoirement énumérables, ils possèdent à la fois des codes $\Delta^0_1$ et $\Sigma^0_1$. La représentation par des codes $\Delta^0_1$ est cependant calculatoirement plus informative, au sens où elle donne accès à plus d'informations sur l'ensemble qu'elle représente. En particulier, la fonction partielle $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ telle que si $e$ est un code $\Delta^0_1$ d'un ensemble $A$, est définie pour $x \in \mathbb{N}$ par $f(e, x)\downarrow = 1$ si $x \in A$ et $f(e, x)\downarrow = 0$ si $x \notin A$, est calculable[1], tandis que la fonction équivalente pour les codes $\Sigma^0_1$ ne l'est pas.

**Exercice 7.5.** Montrer qu'il n'existe pas de fonction $f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ qui soit totale, calculable et telle que $f(e, x) = 1$ si $x \in W_e$ et $f(e, x) = 0$ si $x \notin W_e$. ♢

De manière générale, on peut représenter tout ensemble de la hiérarchie arithmétique par des entiers à l'aide du théorème de Post.

**Définition 7.6.** Un *code* $\Sigma^0_{n+1}$ (resp. $\Delta^0_{n+1}$) d'un ensemble $A$ est un entier $e$ tel que $W_e(\emptyset^{(n)}) = A$ (resp. $\Phi_e(\emptyset^{(n)}) = A$). ♢

**Remarque**

Étant donné un ensemble $A$, on aura tendance à privilégier le type de code qui apporte le plus d'information calculatoire sur $A$. Ainsi, si $A$ est calculable, il est préférable de manipuler un code $\Delta^0_1$ plutôt que $\Sigma^0_1$.

Cette recommandation s'applique également à des concepts de calculabilité plus élaborés, comme les ensembles low. Pour rappel, un ensemble $A$ est low si $A' \leqslant_T \emptyset'$. Comme $A \leqslant_T A' \leqslant_T \emptyset'$, les ensembles low sont en particulier $\Delta^0_2$, et peuvent donc être représentés par un code $\Delta^0_2$, c'est-à-dire un entier $e$ tel que $\Phi_e(\emptyset') = A$. Cependant, cette représentation perd de l'information spécifique aux ensembles low (voir l'exercice 7.11), comme la capacité de $\emptyset'$ de décider $A'$. Il est donc préférable de représenter l'ensemble $A$ par un code $e$ tel que $\Phi_e(\emptyset') = A'$.

**Définition 7.7.** Un *code de lowness* d'un ensemble $A$ est un entier $e$ tel que $\Phi_e(\emptyset') = A'$. ♢

1. Si $e$ n'est pas un code $\Delta^0_1$, la fonction $f$ agit tout de même comme si c'était le cas, et l'on aura éventuellement alors $f(e, x)\uparrow$.

Enfin, dans le cas des ensembles finis, il est possible de stocker plus d'information que le simple fait d'être calculable. En effet, étant donné une suite $e_0, e_1, \ldots$ de codes $\Delta^0_1$ d'ensembles finis, il n'est pas possible de calculer uniformément le cardinal de ces ensembles (voir l'exercice 7.10). On préférera donc la notion de code canonique qui contient notamment l'information de la taille de l'ensemble.

**Définition 7.8.** Le *code canonique* d'un ensemble fini $F$ est l'entier naturel $\sum_{i \in F} 2^i$. ♢

On vérifie sans peine via l'exercice suivant qu'un code canonique contient toute l'information d'un ensemble fini, y compris la possibilité de connaître son dernier élément.

**Exercice 7.9.** Soit $D_0, D_1, \ldots$ la suite des ensembles finis telle que $D_n$ est de code canonique $n$.

1. Montrer que la fonction $f : \mathbb{N} \to \mathbb{N}$ définie par $f(n) = |D_n|$ est calculable.
2. Montrer que la fonction $g : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ définie par $g(n, x) = 1$ ssi $x \in D_n$ est calculable. ◇

Les deux exercices suivants permettent de montrer que les informations que nous donnent les codes canoniques d'ensembles finis ou les codes de lowness ne peuvent être obtenues via des codes calculables ou $\Delta^0_2$.

**Exercice 7.10.** (⋆⋆) Supposer par l'absurde qu'il existe une fonction partielle calculable $f : \mathbb{N} \to \mathbb{N}$ telle que si $e$ est un code $\Delta^0_1$ d'un ensemble fini, alors $f(e)$ renvoie la taille de cet ensemble. En utilisant le théorème du point fixe, montrer qu'il existe $a$ un code $\Delta^0_1$ d'un ensemble fini tel que la taille de cet ensemble est différente de $f(a)$. ◇

**Exercice 7.11.** (⋆⋆) Supposer par l'absurde qu'il existe une fonction partielle calculable $f : \mathbb{N} \to \mathbb{N}$ telle que si $e$ est un code $\Delta^0_2$ d'un ensemble low $X$, alors $f(e)$ renvoie un code $\Delta^0_2$ pour $X'$. En utilisant le théorème du point fixe, montrer qu'il existe $a$ un code $\Delta^0_2$ d'un ensemble calculable $X$ tel que $f(a)$ est le code $\Delta^0_2$ d'un ensemble différent de $X'$. ◇

Chapitre 6

# La thèse de Church-Turing

## 1. L'Entscheidungsproblem et la quête du Graal

En l'an 1928, dans une période troublée par de profondes remises en questions sur les fondements des mathématiques, David Hilbert et Wilhelm Ackermann posèrent la question de l'existence d'un algorithme permettant de décider de la validité de n'importe quel énoncé mathématique. Ici, le mot *algorithme* est à prendre dans un sens large, pour désigner un ensemble d'étapes de calcul élémentaires que peut réaliser un mathématicien. Ce défi lancé aux logiciens et passé à la postérité sous le nom d'*Entscheidungsproblem* — problème de décision — marque le début d'une longue quête fondationnelle sur la formalisation de la notion d'algorithme.

Les théorèmes d'incomplétude de Gödel [1] prouvés en 1931, énonçant l'existence d'énoncés fondamentalement indécidables dans toute théorie raisonnable permettant de formaliser l'arithmétique, furent particulièrement malvenus dans une période qui cherchait à initier une nouvelle vague d'optimisme et de confiance dans les mathématiques. Ils eurent aussi comme conséquence de faire pencher la balance vers l'existence d'une solution négative à l'Entscheidungsproblem.

Une réponse positive à l'Entscheidungsproblem aurait été un algorithme ou une série d'étapes déterministes, permettant de démontrer tout énoncé mathématique ou sa négation. Une réponse négative consiste quant à elle en la preuve qu'une telle méthode systématique ne peut exister. Cette direction pose un tout autre problème, à savoir définir formellement le concept

1. Voir le chapitre 9

d'algorithme, ou de manière à peu près équivalente, de trouver une formalisation robuste et consensuelle de la notion de fonction calculable.

Il convient de préciser que le premier ordinateur, l'ENIAC, a été construit en 1940, soit une décennie après la formulation de l'Entscheidungsproblem. La notion de fonction effectivement calculable ne fait donc pas référence à ce qui est calculable par un ordinateur, mais par l'esprit humain. Il s'agissait de trouver un procédé systématique, ou algorithme au sens informel du terme, permettant à un mathématicien de répondre à toute question mathématique.

Plusieurs définitions furent proposées au cours des années qui suivirent, chacune conjecturée de manière plus ou moins convaincante comme capturant exactement la notion épistémologique de fonction effectivement calculable. Ces définitions furent assez rapidement prouvées équivalentes, mais peinèrent à convaincre la communauté scientifique de leur capacité à capturer toutes les fonctions effectivement calculables. Ce n'est qu'en 1936 qu'Alan Turing amena un consensus en présentant son modèle de calcul, la *machine de Turing*, avec une démonstration éclatante de son équivalence avec les autres modèles, en particulier avec celui utilisé par Gödel pour montrer son fameux théorème d'incomplétude, apportant ainsi une réponse définitive à l'Entscheidungsproblem. Le lecteur intéressé par l'histoire de la calculabilité trouvera un excellent chapitre dédié au sujet dans l'ouvrage *Turing computability : Theory and Applications* de Robert Soare.

Bien que les différents modèles de calcul aient été depuis prouvés équivalents, nous allons détailler les principaux parmi ceux qui marquèrent cette histoire, chacun présentant son propre intérêt, en mettant l'accent sur un aspect différent de la notion de fonction calculable. Dans ce qui suit, nous ne considérerons que des fonctions partielles, de $\mathbb{N}^n$ vers $\mathbb{N}$ pour $n \geqslant 1$. On écrira $g(\overline{x})\downarrow = y$ pour signifier que $g$ est définie sur $\overline{x} \in \mathbb{N}^n$ et vaut $y$ ; la notation $\overline{x}$ étant un raccourci pour $x_1, \ldots, x_n$.

**Les fonctions générales récursives**. Les fonctions récursives furent introduites sous une forme restreinte par Gödel dans le cadre de ses théorèmes d'incomplétude en 1931. Cette classe de fonctions a été généralisée par Gödel et Herbrand en 1934 pour obtenir les fonctions générales récursives, capturant d'après la thèse de Church-Turing — thèse détaillée dans la section suivante — l'intégralité des fonctions calculables.

L'idée sous-tendant la définition des fonctions générales récursives est très simple : partir de quelques fonctions élémentaires, dont la calculabilité ne fait pas place au doute, puis les combiner pour obtenir de nouvelles fonctions plus complexes, toujours calculables. Quelles sont les combinaisons valides permettant de créer de nouvelles fonctions ? Si deux fonctions sont

calculables, leur composition devrait naturellement être calculable : il suffit d'exécuter en série les étapes de calcul de chacune des fonctions. De manière un peu moins évidente, les fonctions définies par récursion à partir de fonctions calculables sont encore calculables. En effet, si l'on définit une fonction $f : \mathbb{N} \to \mathbb{N}$ par $f(0) = v$ pour un entier $v$, et $f(n+1) = g(n, f(n))$ pour une fonction calculable $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, pour obtenir la valeur de $f(n)$, il suffit de calculer successivement $f(1), f(2), \ldots, f(n)$ à l'aide des valeurs précédentes et en suivant les étapes de calcul de la fonction $g$. Enfin, si une fonction $g : \mathbb{N} \to \mathbb{N}$ est calculable, la recherche de la plus petite entrée $n$ telle que $g(n) = 0$ est également calculable, via une boucle qui incrémente $n$ jusqu'à avoir $g(n) = 0$ (cette recherche pouvant ne jamais aboutir).

**Définition 1.1.** La classe $\mathcal{C}$ des *fonctions générales récursives* est la plus petite classe de fonctions partielles contenant les fonctions de base suivantes :

(a) La fonction successeur : $\mathrm{succ}(x) = x + 1$

(b) Les fonctions constantes : $c_m^n(x_1, \ldots, x_n) = m$ pour tous $n, m \geqslant 0$

(c) Les projections : $p_i^n(x_1, \ldots, x_n) = x_i$ pour tout $n$ et tout $i \in 1, \ldots, n$

et qui est close par les opérations suivantes.

(i) Composition : si $g_1, \ldots, g_m \in \mathcal{C}$ sont des fonctions de $n$ variables et $h \in \mathcal{C}$ est une fonction de $m$ variables, alors la fonction

$$f(\overline{x}) = h(g_1(\overline{x}), \ldots, g_m(\overline{x}))$$

est dans $\mathcal{C}$.

(ii) Récursion primitive : si $g, h \in \mathcal{C}$ sont des fonctions partielles de respectivement $n$ et $n+2$ variables, la fonction $f$ définie par $f(\overline{x}, 0) = g(\overline{x})$ et $f(\overline{x}, m+1) = h(\overline{x}, m, f(\overline{x}, m))$ est alors une fonction partielle de $n+1$ variables dans $\mathcal{C}$.

(iii) Minimisation : si $g \in \mathcal{C}$ est une fonction partielle de $n+1$ variables, alors la fonction partielle $f$ qui à $\overline{x}$ associe le plus petit entier $m$, s'il existe, tel que $g(\overline{x}, i) \downarrow$ pour tout $i \leqslant m$ et tel que $g(\overline{x}, m) = 0$, est dans $\mathcal{C}$. Si $m$ n'existe pas, alors $f$ n'est pas définie sur $\overline{x}$.

La classe des *fonctions primitives récursives* est la plus petite classe de fonctions totales contenant les fonctions de (a),(b),(c) et close par les schémas (i) et (ii) de composition et récursion primitive. ♢

Les fonctions générales récursives présentent l'avantage de mettre en valeur les propriétés de clôture de la notion de fonction calculable, à commencer par la clôture par composition. Notons que les fonctions primitives récursives sont totales, contrairement aux fonctions générales récursives, pour lesquelles le schéma de minimisation introduit une possibilité de par-

tialité. Intuitivement, son implémentation en programmation consisterait en une boucle `while` cherchant exhaustivement un entier $m$ satisfaisant la propriété. Si cet entier n'existe pas, l'exécution du programme ne sortira jamais de la boucle, et le programme ne s'arrêtera pas. Nous étudierons ce modèle de manière détaillée dans la section 3 en nous efforçant de montrer qu'il correspond bien à la notion de fonction effectivement calculable. Mentionnons avant cela deux autres modèles de calcul.

**Le $\lambda$-calcul**. Défini par Church dans les années 30, le $\lambda$-calcul est un formalisme minimaliste servant de fondement théorique aux langages de programmation. Contrairement aux fonctions générales récursives, qui manipulent deux types d'objets, à savoir les entiers et les fonctions sur les entiers, le $\lambda$-calcul n'a qu'un seul objet primitif : les $\lambda$-*fonctions*. Celles-ci ne prennent pas en paramètre des entiers, mais des $\lambda$-fonctions. Il faudra donc recourir au codage des entiers par des $\lambda$-fonctions pour définir une notion de fonction calculable sur les entiers.

Les $\lambda$-fonctions sont définies par un langage d'expressions, comme c'est souvent le cas en mathématiques. Par exemple, $x, y \mapsto x + y$ est une expression définissant la somme de deux entiers. Cependant, en $\lambda$-calcul, les paramètres étant eux-mêmes des $\lambda$-fonctions, les seules opérations valides sont celles de manipulation de fonctions, à savoir, l'ajout d'un paramètre à une fonction, et l'application d'une fonction à ses paramètres. Par exemple, $f \mapsto (g \mapsto f(g))$ définit la fonction qui prend en paramètre une fonction $f$, et renvoie la fonction qui prend en paramètre une fonction $g$, et renvoie le résultat de l'application de $f$ à $g$.

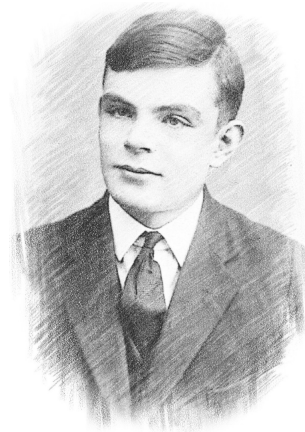
L'aspect minimaliste du $\lambda$-calcul facilite les raisonnements sur le formalisme en lui-même, mais demande beaucoup plus de travail pour définir des fonctions non triviales sur les entiers. Il est notamment plus difficile de se convaincre que toutes les fonctions calculables peuvent-être représentées par des $\lambda$-fonctions. Comme exprimé précédemment, il est nécessaire de fixer une convention pour représenter les entiers. Il paraît assez naturel d'identifier l'entier $n$ avec la $\lambda$-fonction prenant en entrée une $\lambda$-fonction $f$ et retournant sa $n$-ième itération. Par exemple, l'entier 0 est représenté par la fonction qui prend en entrée une fonction $f$, et renvoie sa 0-ième itération, autrement dit renvoie la fonction identité. Dans le formalisme du $\lambda$-calcul, elle s'écrit $f \mapsto (x \mapsto x)$. De la même manière, l'entier 2 correspond à la fonction $f \mapsto (x \mapsto f(f(x))$. Appelons $\overline{n}$ la $\lambda$-fonction représentant l'entier $n$ ; une fonction $g : \mathbb{N} \to \mathbb{N}$ est $\lambda$-*définissable* s'il existe une $\lambda$-fonction $h$ qui à $\overline{n}$ associe $\overline{g(n)}$.

La syntaxe du $\lambda$-calcul est assez absconse au premier abord, et demande un peu de manipulation pour se familiariser avec les concepts. De nos jours, de nombreuses variantes et enrichissements sont étudiés afin de fournir un

fondement théorique aux langages de programmation fonctionnelle. Il s'agit d'un domaine de recherche très actif.

**Les machines de Turing**. Si le $\lambda$-calcul fournit une base théorique aux langages de programmation, les machines de Turing peuvent être vues comme précurseurs de l'ordinateur moderne.

Conçue en 1936 par Alan Turing, cette machine est inspirée de la machine à écrire de son père. Elle comporte un *ruban*, qui peut être vu comme une succession de *cases* indexées par des entiers. On peut faire l'analogie avec des bits auxquels on accède dans la mémoire d'un ordinateur via un système d'adressage. La machine dispose aussi d'une tête de lecture/écriture qui peut se déplacer d'une case à l'autre, puis en lire ou en modifier le contenu. La machine va s'exécuter en fonction d'une entrée $n$. Au début du calcul, la tête de lecture se trouve au début du ruban, dont les $n$ premières cases sont initialisées à 1, tandis que les cases restantes valent 0.



Alan Turing, 1912–1954

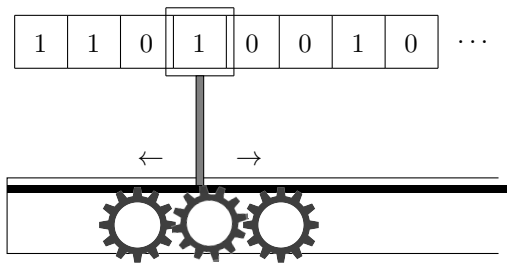



FIGURE 1.3 – *Représentation d'une machine de Turing, avec sa tête de lecture/écriture se déplaçant sur les cases du ruban*

La machine possède également un nombre fini d'*états*, incluant un état de départ dans lequel elle se trouve au début du calcul, ainsi qu'un état d'arrêt qui indique la fin du calcul lorsque la machine l'atteint. Les déplacements de la tête de lecture, le remplacement de la valeur de la case courante, ainsi que les changements d'état, sont soumis à un ensemble d'instructions représentées comme suit : étant donné l'état courant de la machine et la

valeur de la case où se trouve la tête de lecture, une règle va décider de l'état suivant de la machine, de changer éventuellement la valeur de la case, et de déplacer la tête d'une case à droite ou à gauche. Une machine de Turing est donc un automate élaboré conçu pour réaliser une tâche spécifique. Dans son article fondateur, Turing a démontré l'existence d'une machine de Turing *universelle* : une machine permettant de simuler le calcul de n'importe quelle autre machine de Turing.

Contrairement aux fonctions générales récursives ou au $\lambda$-calcul, le modèle de Turing constitue un processus mécanique très concret. On peut de fait réellement construire une machine de Turing universelle. Ce modèle a notamment l'avantage de rendre explicite la notion d'étape atomique de calcul ainsi que de quantifier l'espace mémoire utilisé. Pour ces raisons, les machines de Turing sont prises comme modèle de référence pour définir la théorie de la complexité algorithmique.

**S'abstraire des modèles**. À l'instar d'une citation de Michael Fellows [2], la calculabilité n'est pas plus l'étude des modèles de calcul que l'astronomie n'est la science des télescopes. La calculabilité s'émancipe très rapidement des détails d'implémentation des modèles de calcul, pour manipuler les programmes de manière abstraite. Il est malgré tout instructif de voir en détail au moins un modèle de calcul, en particulier pour se forger une intuition quand celle-ci fait défaut, et c'est ce que nous ferons très bientôt dans la section 3.

## 2. Thèse de Church-Turing

En 1934, face au succès du $\lambda$-calcul, Church soumit l'idée à Gödel selon laquelle les fonctions $\lambda$-définissables captureraient la notion de fonction effectivement calculable, laissant Gödel dubitatif. Avec le développement des fonctions générales récursives de Herbrand et Gödel la même année et la preuve de leur équivalence avec le $\lambda$-calcul, Church formula publiquement sa thèse, connue sous le nom de *thèse de Church*, affirmant que les fonctions générales récursives coïncidaient avec les fonctions effectivement calculables. Son argumentation ne convainc pas Gödel, bien qu'il fût l'un des instigateurs des fonctions générales récursives. Avec son modèle de machine éponyme, Alan Turing fit finalement consensus en 1936, en démontrant ce dernier équivalent au modèle du $\lambda$-calcul et à celui des fonctions générales récursives, ce qui conduisit à ce que l'on appelle aujourd'hui la thèse de Church-Turing.

2. ≪ L'informatique n'est pas plus l'étude des ordinateurs, que l'astronomie n'est celle des télescopes. ≫ [59]

**Thèse 2.1 (Church-Turing).**
Les fonctions effectivement calculables sont celles calculables par une machine de Turing, ou de manière équivalente les fonctions générales récursives ou encore les fonctions $\lambda$-définissables. ∎

Si Church fut le premier à formuler la thèse selon laquelle les fonctions $\lambda$-définissables correspondaient aux fonctions effectivement calculables, on attribue généralement la paternité de la calculabilité à Turing. La thèse de Church-Turing n'étant pas un énoncé mathématique, il n'est pas possible de la prouver formellement. Elle peut néanmoins être validée par ce qui se rapproche le plus d'une preuve au sens social du terme, c'est-à-dire par un argument engendrant un consensus de la communauté scientifique. C'est ce à quoi Turing est parvenu par la démonstration suivante.

**La démonstration de Turing.** Pour justifier sa thèse, Turing a employé trois types d'arguments :

(1) une description du processus par lequel un mathématicien effectue un calcul et sa formalisation par une machine de Turing,

(2) la preuve de l'équivalence des machines de Turing avec les modèles de calculs existants,

(3) le développement explicite de grandes classes de fonctions calculables par les machines de Turing. Voici les grandes lignes du premier argument de Turing :

Considérons un mathématicien, M. Dupont, effectuant un calcul. M. Dupont possède un crayon et une quantité potentiellement illimitée de papier. Étant donné la précision finie de son crayon, chaque feuille ne peut contenir qu'un nombre fini de symboles. Par simplification, et sans perte de généralité, on peut considérer que chaque feuille est une case d'un ruban infini, ne contenant qu'un seul symbole appartenant à un alphabet fini suffisamment grand. Le processus de calcul est le suivant : alors qu'il se trouve dans un état mental $e_0$, M. Dupont se situe face à la feuille courante, dans son champ de vision. Il lit les notes, avant de les corriger, en effaçant et changeant le symbole écrit sur la feuille. Cette lecture va faire évoluer ses pensées, et il se retrouvera dans un état mental $e_1$. Il va potentiellement tourner la page, ou revenir en arrière pour relire des notes précédentes, jusqu'à arriver à la fin de son calcul. M. Dupont considérera alors son calcul terminé, et se retrouvera dans l'état mental correspondant que l'on appellera *état final*.

**Du bon usage de la thèse de Church-Turing.** Il convient de se faire une idée claire de ce que dit la thèse de Church-Turing, de ses limites et de son usage. La thèse de Church-Turing n'est ni un théorème, ni une conjecture. Elle ne peut être formellement prouvée ou réfutée, par le simple fait qu'elle

n'est pas un énoncé mathématique, mais plutôt un pont entre un concept mathématique et une notion épistémologique. Cette thèse n'est pas pour autant une affirmation arbitraire, car elle est étayée par un raisonnement qui peut être soumis à la critique.

Si la thèse de Church-Turing peut être remise en question, voire un jour majoritairement rejetée, le développement de la calculabilité n'en repose pas moins sur des bases solides, indépendantes de cette correspondance. L'équivalence entre les fonctions calculables par les machines de Turing, les fonctions générales récursives, les fonctions $\lambda$-définissables, et les fonctions programmables en C, Java ou Python, est bien un théorème qui ne dépend pas de la thèse de Church-Turing. S'il est fréquent en calculabilité de décrire de manière informelle un algorithme pour ensuite en déduire l'existence d'une machine de Turing l'implémentant, ce procédé n'est pas à strictement parler un appel à la thèse de Church-Turing. Il s'agit plutôt d'une preuve informelle permettant de convaincre l'interlocuteur que, si nécessaire, il serait aisé de programmer cet algorithme dans un quelconque langage de programmation.

En outre, si la thèse de Church-Turing venait à être invalidée, l'édifice conceptuel construit autour de la calculabilité resterait, et continuerait vraisemblablement à être étudié. Il existe une hiérarchie de langages formels et de modèles de calcul, appelée *hiérarchie de Chomsky*. On y trouve par exemple les *langages rationnels*, correspondant aux langages reconnus par une classe de machines appelées *automates finis*. Bien que ces modèles soient pour la plupart moins expressifs que les machines de Turing, ils n'en sont pas moins un sujet très actif de recherche de nos jours. Si l'on trouve un jour de nouveaux paradigmes de calcul, la notion existante de fonction calculable n'en demeurera pas moins une classe de fonctions très robuste, et continuera à être étudiée au même titre que les langages rationnels ou tout autre niveau de la hiérarchie de Chomsky.

Certains phénomènes naturels sont étudiés dans l'espoir de résoudre des problèmes non calculables. Ces notions de calculabilité sont réunies sous le nom d'*hypercalcul* (nous en verrons une approche formelle dans la partie IV). Il n'existe pas à ce jour de perspective de réalisation de tels calculs. La découverte de nouveaux phénomènes de calcul dans la nature n'invaliderait cependant probablement pas la thèse de Church-Turing, car ils auraient peu de chances de satisfaire la définition de fonction effectivement calculable d'après Rosser [184], c'est-à-dire « une méthode dont chaque étape est précisément prédéterminée et qui produira de manière certaine une réponse en un nombre fini d'étapes. »

# 3. Étude détaillée des fonctions récursives

L'objectif de cette section est de convaincre le lecteur que les fonctions générales récursives coïncident avec les fonctions effectivement calculables telles que définies informellement dans les sections 3-2 et 3-1, comme étant « les fonctions que l'on peut programmer ». Les intérêts d'une telle étude sont multiples. Elle permet en premier lieu d'avoir une définition mathématique précise de ce qu'est une fonction calculable : il s'agit sans perte de généralité des fonctions générales récursives. Ensuite, l'étude que nous donnons fournira du même coup une preuve mathématique de l'existence d'une machine universelle telle que définie dans le théorème 3-3.1, concept utilisé tout au long de cet ouvrage. Pour finir, notre étude isolera la notion de fonction primitive récursive comme une sous-classe stricte des fonctions calculables ; outre une certaine importance épistémologique (voir le théorème 3.22), cette sous-classe présente un indéniable intérêt en logique mathématique. Nous en verrons un exemple d'utilisation dans l'étude des mathématiques à rebours, à la section 23-7.

### 3.1. Machines à registres et diagrammes de programmation

Pour nous convaincre que les fonctions générales récursives coïncident avec les fonctions calculables, nous partons d'un modèle de calcul proche des langages de programmation modernes : *les programmes structurés*, exécutés par des *machines à registres*. Nos développements reprennent ici dans les grandes lignes le cours de calculabilité d'A. Durand et P. Rozière [51] du Master de logique mathématique de l'université Paris Diderot.

**Les machines à registres**. La littérature scientifique en a décliné plusieurs versions, sous le nom de « random access machine » (Melzak [155], Lambek [135], Shepherdson and Sturgis [197], Peter [174], Elgot et Robinson [55]). Il s'agit de machines dites à « random access memory » ou RAM, nom générique aujourd'hui pour désigner la mémoire vive des ordinateurs. Le terme « random access » (accès aléatoire) doit être compris par opposition à « sequential access » (accès séquentiel), et fait référence au fait que l'on peut accéder à chaque case mémoire directement à partir de son adresse, contrairement par exemple au modèle des machines de Turing, pour lesquelles une tête de lecture doit se déplacer case après case dans la mémoire pour arriver à l'endroit désiré.

La mémoire d'une machine à registre sera toutefois plus élémentaire qu'une mémoire RAM moderne : il s'agit simplement d'un nombre fini de *registres* $R_0, R_1, \ldots, R_k$ pour $k \in \mathbb{N}$ arbitraire. Chaque registre peut contenir un entier quelconque positif ou nul. Notons que les entiers peuvent être arbitrairement grands, et donc que chaque registre représente un espace mémoire « non borné ».

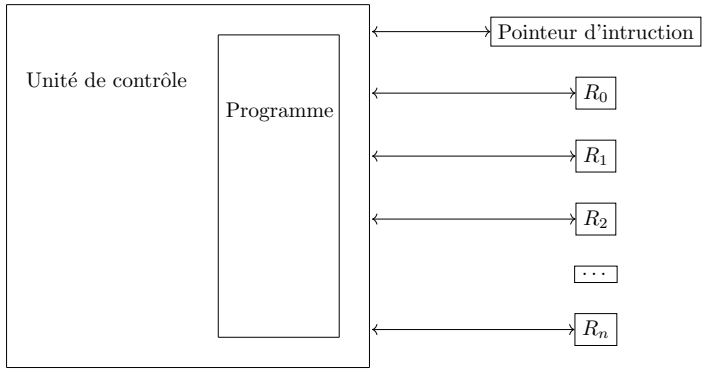



FIGURE 3.1 – *Le modèle des machines à registres*

**Les programmes structurés**. Une machine à registre va exécuter un programme qui consiste en une liste finie d'instructions permettant de faire des calculs. Il existe de nombreuses variantes possibles quant au jeu d'instruction que l'on s'autorise. Nous en présentons un, volontairement proche de celui d'un langage de programmation impérative moderne.

**Définition 3.2.** Un *programme structuré* peut contenir les instructions suivantes.

1. ▷ Incrémentation d'un registre : $\ll R_i := R_i + 1 \gg$.
   ▷ Décrémentation d'un registre : $\ll R_i := R_i - 1 \gg$.
   ▷ Assignement d'un registre : $\ll R_i := x \gg$, $x \in \mathbb{N}$, ou $\ll R_i := R_j \gg$.
   *Ces instructions respectivement incrémentent la valeur de $R_i$ de 1, la décrémentent de 1 (sauf si la valeur est de 0, auquel cas rien ne se passe), mettent la valeur de $R_i$ à $x$ ou la mettent à celle de $R_j$).*
2. L'instruction conditionnelle : $\ll$ if $R_i = 0$ then $S$ else $S' \gg$, où les suites finies $S = (S_0, \ldots, S_n)$ et $S' = (S'_0, \ldots, S'_m)$ sont des suites d'instructions structurées.
   *Chaque instruction de $S$ est exécutée séquentiellement si $R_i$ est égal à 0. Sinon, chaque instruction de $S'$ est exécutée séquentiellement.*
3. L'instruction de boucle for : $\ll$ for $i = 1$ to $R_i$ do $S \gg$, où la suite finie $S = (S_0, \ldots, S_n)$ est une suite d'instructions structurées.
   *Soit $N$ le nombre présent dans le registre $R_i$ au moment où le programme commence cette instruction. Chaque instruction de $S$ est exécutée séquentiellement, le tout $N$ fois. Notez que si la valeur de $R_i$ change pendant l'exécution des instructions de $S$, cela ne change pas le nombre de fois que la boucle s'effectue.*

4. L'instruction de boucle while : « while $R_i \neq 0$ do $S$ » où la suite finie $S = (S_0, \ldots, S_n)$ est une suite d'instructions structurées.

   *Chaque instruction de $S$ est exécutée séquentiellement tant que le registre $R_i$ est différent de 0.*

L'exécution d'un programme structuré s'arrête une fois la dernière instruction exécutée. ♢

Notons qu'un programme structuré n'a qu'un nombre fini d'instructions et ne peut donc utiliser qu'un nombre fini de registres.

**Boucle `for` vs Boucle `while`**

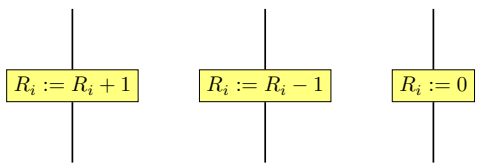
Le lecteur remarquera peut-être que l'instruction de boucle `for` est redondante dans la mesure où elle peut toujours être remplacée par une instruction de boucle `while`. Nous verrons que l'inverse n'est pas vrai. En particulier, le nombre de fois qu'une boucle `for` s'exécute est nécessairement fini, ce qui n'est pas le cas des boucles `while`, au sein desquelles un calcul peut se retrouver bloqué dans ce que l'on appelle classiquement en programmation *une boucle infinie*. Nous verrons que l'instruction de boucle `while` est indispensable, et que certaines fonctions calculables (et totales) ne peuvent se programmer en n'utilisant que des boucles `for`.

**Les diagrammes de programmation**. La programmation structurée trouve sa genèse dans les travaux de Goldstine et von Neumann [75], qui manifestent, dès 1946, un souci non plus de capturer mathématiquement la notion de calcul, mais celui de développer un système de programmation. Ils développent une manière de présenter des algorithmes à base de *diagrammes de programmation*.

Nous en donnons ici la présentation simplifiée que l'on trouve dans l'ouvrage de référence de Piergiorgio Odifreddi [169].

**Définition 3.3.** Un diagramme de programmation est obtenu en connectant entre elles des briques de base des deux types suivants.

▷ Des instructions d'assignements :



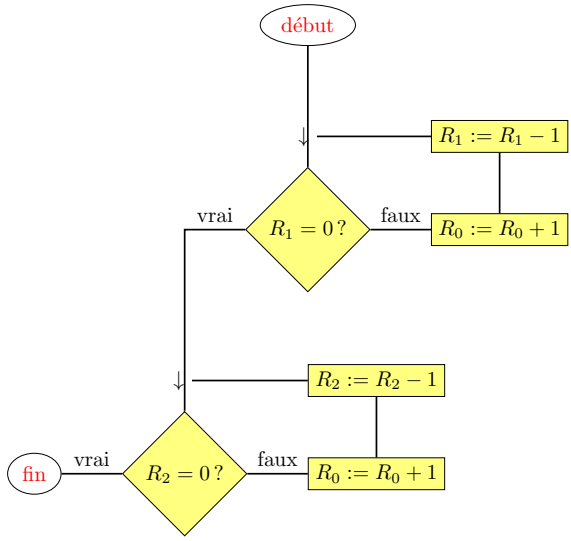
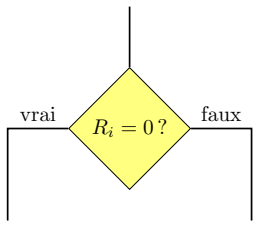



FIGURE 3.4 – *Un diagramme de programmation pour la fonction d'addition de deux nombres. On suppose que le calcul démarre avec $R_0 = 0$, $R_1 = n_1$ et $R_2 = n_2$. À la fin de l'exécution, on aura $R_0$ égal à $n_1 + n_2$.*

▷ Une instruction conditionnelle :




Un diagramme de programmation possède une entrée et une ou plusieurs sorties. Le calcul s'effectue de manière linéaire en exécutant chaque bloc depuis l'entrée vers d'une de ses sorties. ♢

À titre d'exemple, la figure 3.4 est un diagramme de programmation correspondant à la fonction d'addition.

Les diagrammes de programmation peuvent être programmés sur des machines à registre avec un jeu d'instructions comportant des sauts conditionnels et inconditionnels (ces derniers étant simplement appelés « sauts »), c'est-à-dire des instructions de type *goto* en anglais, qui permettent de déterminer quelle est la prochaine instruction du programme qui sera exécutée. Il s'agit encore aujourd'hui du mécanisme à l'œuvre dans les différents langages assembleurs des micro-processeurs.

**Définition 3.5.** Un programme *goto* est une suite numérotée d'instructions $I_0, I_1, \ldots, I_n$ où chaque instruction est de l'un des types suivants :

1. ▷ Incrémentation d'un registre : $\ll R_i := R_i + 1 \gg$.
   ▷ Décrémentation d'un registre : $\ll R_i := R_i - 1 \gg$.
   ▷ Assignement d'un registre à 0 : $\ll R_i := 0 \gg$.
2. L'instruction de saut conditionnel : $\ll$ if $R_i = 0$ goto $n_1$ else goto $n_2 \gg$.
   *Si $R_i$ est égal à 0, l'instruction numéro $n_1$ est la prochaine à être exécutée, sinon c'est l'instruction numéro $n_2$.*
3. Le saut inconditionnel : $\ll$ goto $m \gg$.
   *L'instruction numéro $m$ est la prochaine à être exécutée.*

L'exécution d'un programme goto s'arrête quand il n'y a plus de prochaine instruction à exécuter (ce qui en particulier peut arriver après une instruction de type $\ll$ goto $m \gg$ dans un programme comportant moins de $m$ instructions.) ♢

**Remarque**

Notons que le saut inconditionnel $\ll$ goto $m \gg$ peut être remplacé par un saut conditionnel $\ll$ if $R_i = 0$ goto $m$ else goto $m \gg$. Un programme *goto* peut donc être exprimé sans l'instruction 3.

Il est clair que les diagrammes de programmation sont interchangeables avec les programmes de type goto, et nous utiliserons un formalisme ou l'autre en fonction de la situation.

**Fonctions calculables**. Les notations $\Phi_e(x_1,\ldots,x_n)\!\downarrow\!=y$ et $\Phi_e(x_1,\ldots,x_n)\!\uparrow$ utilisées tout au long du livre s'appliquent naturellement aux programmes exécutés par des machines à registres, une fois les conventions de passage des paramètres et de récupération du résultat fixées.

**Définition 3.6.** Étant donné $P$ un programme de type structuré ou bien goto, on écrit $P(x_1, \ldots, x_k)$ pour désigner l'exécution de $P$ avec les registres $R_1, \ldots, R_k$ initialisés à $x_1, \ldots, x_k$, et tous les autres registres initialisés à 0. On écrit $P(x_1, \ldots, x_k) \downarrow = x$ pour signifier qu'une telle exécution s'arrête avec la valeur $x$ dans le registre $R_0$, et $P(x_1, \ldots, x_k) \uparrow$ pour signifier que l'exécution ne s'arrête pas. ♢

On peut à présent utiliser notre modèle de calcul pour donner une définition mathématique précise de fonction calculable. Officialisons auparavant les notations dom $f$ et Im $f$, qui désignent respectivement le domaine et l'image d'une fonction $f$.

**Notation**

Étant donné une fonction (éventuellement partielle) $f : A \to B$, on note $\operatorname{dom} f$ le domaine de définition de $f$, et

$$\operatorname{Im} f = \{X \in B \ : \ \exists Y \in \operatorname{dom} f \ f(Y) = X\}$$

son image.

**Définition 3.7.** Une fonction (éventuellement partielle) $f : \mathbb{N}^n \to \mathbb{N}$ est *calculable par programme structuré* (resp. *par programme goto*) s'il existe un programme structuré (resp. un programme goto) $P_f$, tel que

$$P_f(x_1, \ldots, x_n) \downarrow = f(x_1, \ldots, x_n)$$

pour tous éléments $x_1, \ldots, x_n \in \operatorname{dom} f$ et tel que $P_f(x_1, \ldots, x_n) \uparrow$ pour tous éléments $x_1, \ldots, x_n \notin \operatorname{dom} f$. ♢

**Justification du modèle de calcul**. Le programmeur ne sera peut-être pas convaincu par le fait que le modèle des machines à registres avec programmes structurés (ou goto) permet effectivement de programmer toutes les fonctions qu'il pourrait écrire dans son langage favori.

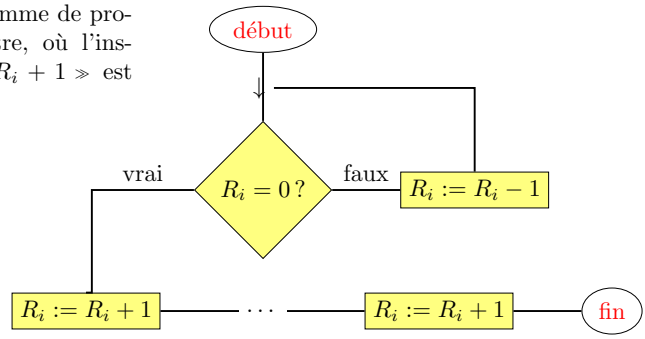
Un aspect en particulier peut susciter l'inquiétude : un langage de programmation moderne permet l'utilisation de tableaux, et même de tableaux dynamiques, dont la taille peut augmenter au fur et à mesure des besoins. Nous verrons par exemple dans la définition 3.24 que la fonction dite d'Ackermann se calcule naturellement à l'aide d'une structure de pile, et qu'il n'est pas du tout clair que l'on puisse s'en passer. Fort heureusement, nous montrerons avec la proposition 3.26 qu'il est parfaitement possible de simuler les manipulations de tableaux dynamiques au sein des machines à registres, en codant ces derniers dans les registres, qui, rappelons-le peuvent contenir des entiers arbitrairement grands.

**Simulation des programmes structurés par les programmes goto**. Nous commençons par montrer que les programmes goto, malgré leur simplicité, sont suffisants pour calculer tout ce que peuvent calculer les programmes structurés.

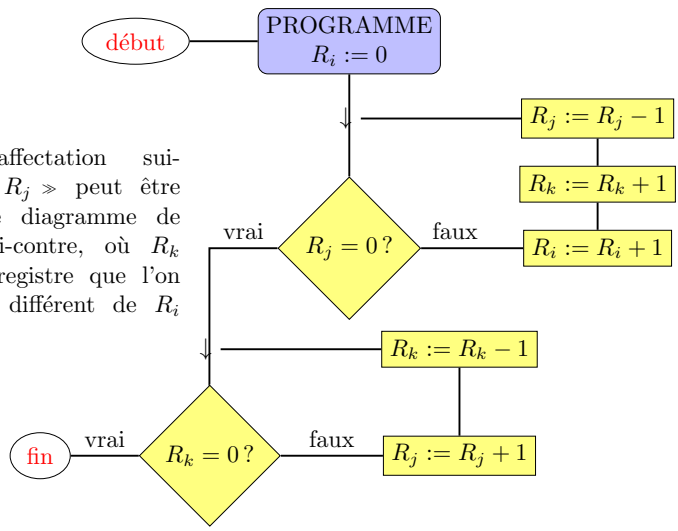
**Proposition 3.8.** Soit $n \in \mathbb{N}^*$, et soit $f : \mathbb{N}^n \to \mathbb{N}$ une fonction calculable par un programme structuré. Alors, la fonction $f$ est calculable par un programme goto. ⋆

Preuve. Il suffit de montrer que chaque instruction d'un programme structuré peut être remplacée par un diagramme de programmation.

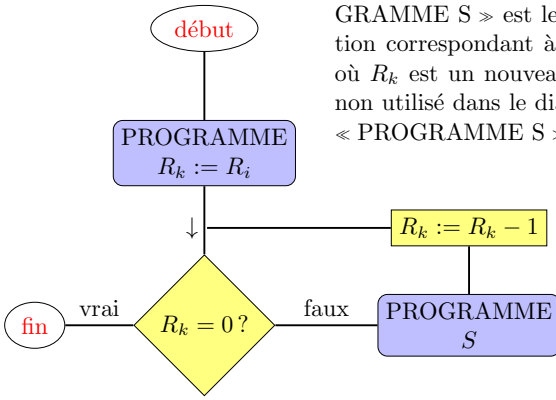
L'instruction d'affectation suivante « $R_i := n$ » peut être remplacée par le diagramme de programmation ci-contre, où l'instruction « $R_i := R_i + 1$ » est répétée $n$ fois.



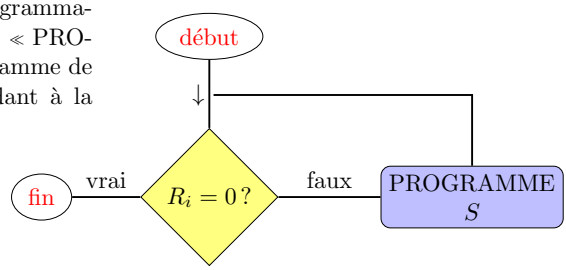

L'instruction d'affectation suivante « $R_i := R_j$ » peut être remplacée par le diagramme de programmation ci-contre, où $R_k$ est un nouveau registre que l'on suppose à 0, et différent de $R_i$ et $R_j$.



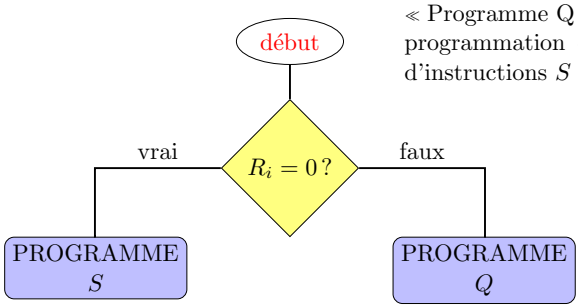
L'instruction de boucle « for $i = 1$ to $R_i$ do $S$ » peut être remplacée par le diagramme de programmation ci-contre, où la boîte « PROGRAMME S » est le diagramme de programmation correspondant à la liste d'instructions $S$, et où $R_k$ est un nouveau registre différent de $R_i$ et non utilisé dans le diagramme de programmation « PROGRAMME S ».




L'instruction de boucle « while $R_i \neq 0$ do $S$ » peut être remplacée par le diagramme de programmation ci-contre, où la boîte « PROGRAMME S » est le diagramme de programmation correspondant à la liste d'instructions $S$.




L'instruction d'affectation « if $R_i = 0$ then $S$ else $Q$ » peut être remplacée par le diagramme de programmation ci-contre, où les deux boîtes « Programme S » et « Programme Q » sont les diagrammes de programmation correspondant aux suites d'instructions $S$ et $Q$.




■

### 3.2. Les fonctions récursives sont calculables.

Les programmes structurés suivent les concepts de la programmation dite *impérative* : des instructions modifiant l'état de la machine sont exécutées les une après les autres. Les fonctions générales récursives suivent quant à elles le paradigme de la programmation dite *fonctionnelle* : un programme est une composition de fonctions mathématiques et un calcul l'évaluation de ces fonctions. Un avantage de la programmation fonctionnelle souvent mis en avant est l'absence d'*effets de bord* : le résultat d'une fonction dépend de ses paramètres et uniquement de ses paramètres (l'état de la machine sur laquelle la fonction s'exécute n'a pas d'incidence). On peut par exemple brancher la sortie d'une fonction sur l'entrée d'une autre sans s'attendre à de mauvaises surprises. À titre de comparaison, la combinaison de programmes structurés $P_f, P_g$ calculant des fonctions $f, g : \mathbb{N} \to \mathbb{N}$ en un programme calculant la fonction $x \mapsto f(g(x))$ demande un peu de travail, afin justement d'éviter les effets de bord.

**Définition 3.9.** Un programme de type structuré ou bien de type goto est *propre* s'il termine son calcul avec tous ses registres — à l'exception de $R_0$ — dans le même état qu'au début du calcul. ♢

**Exercice 3.10.** Montrer que pour tout programme structuré $M$, il existe un programme structuré $N$ propre tel que $M(\overline{x}) \downarrow = y \leftrightarrow N(\overline{x}) \downarrow = y$. ⋄

**Théorème 3.11 (Wang [229], Peter [175], Ershov [57])**
*Toute fonction générale récursive (éventuellement partielle) est calculable par un programme structuré. Toute fonction primitive récursive est calculable par un programme structuré sans boucle* `while`.

Preuve. On laisse au lecteur le soin de montrer que les fonctions constantes, les fonctions de projection et la fonction successeur sont toutes calculables par un programme structuré. Il reste à traiter les cas suivants.

*Schéma de composition.* Soit

$$f(x_1, \ldots, x_m) = g\big(h_1(x_1, \ldots, x_m), \ldots, h_k(x_1, \ldots, x_m)\big),$$

pour des fonctions $g : \mathbb{N}^k \to \mathbb{N}$ et $h_1, \ldots, h_k : \mathbb{N}^m \to \mathbb{N}$ calculables par des programmes structurés $G, H_1, \ldots, H_m$. Par l'exercice 3.10, on peut supposer que $G, H_1, \ldots, H_m$ sont propres. Supposons que chacun de ces programmes utilise au plus les registres $R_0, \ldots, R_z$ pour $z > m$. Le programme $F$ pour calculer $f$ est donné par la suite d'instructions suivantes.

Programme $F$

⟨Instructions de $H_1$⟩
$R_{z+1} := R_0$
$R_0 := 0$
⟨Instructions de $H_2$⟩
$R_{z+2} := R_0$
$R_0 := 0$
...
⟨Instructions de $H_k$⟩
$R_{z+k} := R_0$
$R_0 := 0$
$R_1 := R_{z+1}$
...
$R_k := R_{z+k}$
⟨Instructions de $G$⟩

Notons que si $G, H_1, \ldots H_m$ n'utilisent pas de boucle `while`, alors le programme pour calculer $F$ n'en utilise pas non plus.

*Schéma de récursion primitive.* Soient

$$\begin{aligned} f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, y+1) &= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{aligned}$$

pour $g$ et $h$ calculables respectivement par des programmes structurés propres $G$ et $H$, utilisant au plus les registres $R_0, \ldots, R_z$ pour $z > n + 2$. Le programme $F$ suivant permet de calculer $f$.

Programme $F$

$R_{z+1} := R_{n+1}$
⟨Instructions de G⟩
$R_{n+1} := 0$
$R_{n+2} := R_0$
**for** $i = 1$ **to** $R_{z+1}$ **do**
  $R_0 := 0$
  ⟨Instructions de H⟩
  $R_{n+1} := R_{n+1} + 1$
  $R_{n+2} := R_0$
**end**

Notons une fois de plus que si $G, H$ n'utilisent pas de boucles `while`, alors le programme pour calculer $F$ n'en utilise pas non plus.

*Schéma de minimisation.* Soit

$$f(x_1,\dots,x_n)=\min\{x\in\mathbb{N} : \forall i\leqslant x\ (g(x_1,\dots,x_n,i)\downarrow\wedge g(x_1,\dots,x_n,x)=0)\},$$

pour $g$ calculable par un programme propre $G$, utilisant au plus les registres $R_0,\dots,R_z$ pour $z > n+1$. Le programme suivant permet de calculer $f$.

Programme $F$

$R_{z+1} := 1$
$R_{n+1} := 0$
**while** $R_{z+1} \neq 0$ **do**
  $R_0 := 0$
  ⟨Instructions de G⟩
  $R_{z+1} := R_0$
  $R_{n+1} := R_{n+1} + 1$
**end**
$R_{n+1} := R_{n+1} - 1$
$R_0 := R_{n+1}$

Cela conclut la preuve. ∎

### 3.3. Étude des fonctions primitives récursives

Nous passons à présent à la partie la plus difficile de ce chapitre. Afin de montrer que les fonctions calculables par programmes structurés sont des fonctions générales récursives, nous avons besoin d'un certain nombre d'outils, notamment sur les fonctions primitives récursives. Nous alternons différentes propositions et exercices permettant de voir qu'il s'agit d'une large classe de fonctions. Rappelons les notations de la définition 1.1 pour les fonctions primitives récursives de bases.

**Notation**

On note $p_i^n : \mathbb{N}^n \to \mathbb{N}$ la projection telle que $p_i^n(x_1,\dots,x_n) = x_i$. On note $c_i^n : \mathbb{N}^n \to \mathbb{N}$ la fonction constante telle que $c_i^n(x_1,\dots,x_n) = i$. On note $\mathrm{succ} : \mathbb{N} \to \mathbb{N}$ la fonction successeur, définie par $\mathrm{succ}(n) = n+1$.

**Exercice 3.12. (⋆)** Montrer que l'addition, la multiplication et la fonction exponentielle sont primitives récursives. ◇

**Exercice 3.13. (⋆)** Montrer que les fonctions prédécesseur et soustraction sont primitives récursives (comme nos fonctions sont à valeur dans $\mathbb{N}$, on utilisera 0 à la place du résultat si celui-ci est négatif). ◇

**Exercice 3.14.** (⋆) Montrer que la fonction $\mathrm{sg} : \mathbb{N} \to \mathbb{N}$ qui à 0 associe 0 et qui à tous les autres entiers associe 1 est primitive récursive. Montrer que la fonction $\overline{\mathrm{sg}} : \mathbb{N} \to \mathbb{N}$ qui à 0 associe 1 et qui à tous les autres entiers associe 0 est primitive récursive. ◇

**Définition 3.15.** Un prédicat $P \subseteq \mathbb{N}^n$ est primitif récursif (resp. général récursif) s'il existe une fonction primitive récursive (resp. générale récursive) $f : \mathbb{N}^n \to \{0,1\}$ telle que $(x_1,\ldots,x_n) \in P \leftrightarrow f(x_1,\ldots,x_n) = 1$ pour tous $x_1,\ldots,x_n \in \mathbb{N}$. ♢

**Exemple 3.16.** Les prédicats de comparaison $\leqslant, <, \geqslant, >, =, \neq$ sont primitifs récursifs via les fonctions suivantes :

$$\begin{array}{lcllcl} a \leqslant b & = & \mathrm{sg}(\mathrm{succ}(b) - a) & a > b & = & b < a \\ a < b & = & \mathrm{sg}(b - a) & a = b & = & (a \leqslant b) \times (b \leqslant a) \\ a \geqslant b & = & b \leqslant a & a \neq b & = & \overline{\mathrm{sg}}(a = b). \end{array}$$

Formellement, les projections sont utilisées si nécessaire, par exemple pour inverser l'ordre des paramètres dans la définition de $\geqslant$ à partir de celle de $\leqslant$.

**Proposition 3.17.** Les fonctions primitives récursives sont closes par définition par cas sur un prédicat primitif récursif : si $g$ et $h$ sont deux fonctions primitives récursives de $\mathbb{N}^p$ dans $\mathbb{N}$, et si $P$ est un prédicat primitif récursif sur $\mathbb{N}^p$, alors la fonction

$$\begin{array}{rcll} f(x_1,\ldots,x_p) & = & g(x_1,\ldots,x_p) & \text{si } P(x_1,\ldots,x_p) \\ & = & h(x_1,\ldots,x_p) & \text{sinon} \end{array}$$

est primitive récursive. ⋆

Preuve. Comme $P$ est un prédicat primitif récursif, il existe une fonction $d : \mathbb{N}^n \to \mathbb{N}$ telle que

$$\begin{array}{rcll} d(x_1,\ldots,x_p) & = & 1 & \text{si } P(x_1,\ldots,x_p) \\ & = & 0 & \text{sinon.} \end{array}$$

On définit donc :

$$\begin{array}{rcl} f(x_1,\ldots,x_p) & = & g(x_1,\ldots,x_p) \times \mathrm{sg}(d(x_1,\ldots,x_p)) \\ & & + h(x_1,\ldots,x_p) \times \overline{\mathrm{sg}}(d(x_1,\ldots,x_p)). \end{array}$$

■

La preuve de la proposition suivante fournit un exemple d'application de la définition par cas.

**Proposition 3.18.** La division entière est primitive récursive. ⋆

PREUVE. On utilise le schéma de récursion primitive couplé au schéma de définition par cas. On définit $\mathrm{div}(a,b) = \mathrm{div}(a,b,a)$, où :

$$\begin{array}{rcll} \mathrm{div}(a,b,0) & = & 0 & \\ \mathrm{div}(a,b,n+1) & = & \mathrm{succ}(n) & \text{si } \mathrm{succ}(n)\times b = a \\ & = & \mathrm{div}(a,b,n) & \text{sinon} \end{array}$$

■

**Exercice 3.19. (⋆)** Montrer que les prédicats primitifs récursifs sont clos par conjonction, disjonction, négation, quantification existentielle bornée et quantification universelle bornée. ◇

**Exercice 3.20. (⋆)** Montrer que les fonctions primitives récursives sont closes par minimisation bornée : si $f : \mathbb{N}^{p+1} \to \mathbb{N}$ est primitive récursive, alors la fonction $g : \mathbb{N}^{p+1} \to \mathbb{N}$ qui à $x_1, \ldots, x_p, n$ associe le plus petit $t \leqslant n$ tel que $f(x_1, \ldots, x_p, t) = 0$ (et 0 si un tel $t \leqslant n$ n'existe pas) est primitive récursive. ◇

Souvenons-nous des bijections de Cantor comme définies dans la section 2-3. Le codage des paires $\langle x_1, x_2\rangle$ est une notation pour l'application de la bijection $\alpha_2 : \mathbb{N}^2 \to \mathbb{N}$ définie par $\alpha_2(x,y) = y + \dfrac{(x+y+1)(x+y)}{2}$. Le codage $\langle x_1, \ldots, x_k\rangle$ des $n$-uplets est une notation pour l'application des bijections $\alpha_k : \mathbb{N}^k \to \mathbb{N}$ définies inductivement par

$$\alpha_{k+1}(x_1, x_2, \ldots, x_{k+1}) = \alpha_2(x_1, \alpha_k(x_2, \ldots, x_{k+1})).$$

**Proposition 3.21.** Pour tout $n$, la bijection

$$x_1, \ldots, x_n \mapsto \langle x_1, \ldots, x_n\rangle$$

est primitive récursive. Pour tout $n$ et tout $i \leqslant n$, la fonction

$$\langle x_1, \ldots, x_n\rangle \mapsto x_i$$

est primitive récursive. ⋆

PREUVE. L'addition, la multiplication et la division entière étant primitives récursives, la fonction $(x,y) \mapsto y + \dfrac{(x+y+1)(x+y)}{2}$ est elle aussi primitive récursive par le schéma de composition. Il en va donc de même pour tout $n$ pour les fonctions $x_1, \ldots, x_n \mapsto \langle x_1, \ldots, x_n\rangle$ qui sont définies inductivement par composition.

La fonction qui à $\langle x_1, x_2\rangle$ associe $x_1$ est primitive récursive, fait qui s'établit en utilisant la minimisation bornée et la clôture des prédicats primitifs

récursifs par quantification existentielle bornée :

$$f(n) = \min\{x \leqslant n \ : \ \exists y \leqslant n \ \langle x, y\rangle = n\}.$$

On laisse au lecteur le soin de broder sur cette idée pour montrer que toutes les projections sont ainsi primitives récursives. ∎

Nous avons à ce stade tous les éléments nécessaires pour montrer un premier théorème important. Nous avions vu avec le théorème 3.11 que les fonctions primitives récursives peuvent être programmées par des programmes structurés n'utilisant pas de boucle `while`. La réciproque est vraie.

**Théorème 3.22**
*Toute fonction calculable par un programme structuré sans boucles* `while` *est primitive récursive.*

Preuve. Pour $k \in \mathbb{N}$ donné, et pour un programme structuré $P$ sans boucles `while` et utilisant au plus les registres $R_0, \ldots, R_k$, on définit la fonction $f_P : \mathbb{N} \to \mathbb{N}$ par $f_P(\langle x_0, \ldots, x_k\rangle) = \langle v_0, \ldots, v_k\rangle$ où $v_i$ est la valeur du registre $R_i$ en fin d'exécution du programme $P$, quand son exécution commence avec ses registres initialisés aux valeurs $x_0, \ldots, x_k$.

Montrons que pour tout programme structuré $P$ sans boucles `while`, la fonction $f_P$ correspondante est primitive récursive. Il est clair que c'est le cas pour le programme vide. Soit $Q$ le programme dont l'unique instruction est $R_i := R_i + 1$. Alors, la fonction $f_Q$ est donnée par

$$f_Q(\langle x_0, \ldots, x_k\rangle) = \langle x_0, \ldots, x_i + 1, \ldots, x_k\rangle.$$

On laisse au lecteur le soin de trouver la fonction primitive récursive correspondant aux instructions $R_i := R_i - 1$, $R_i := c$ pour $c \in \mathbb{N}$ et $R_i := R_j$.

Supposons à présent que la proposition est vraie pour des programmes $P, P'$, via des fonctions $f_P, f_{P'}$, et soit $Q$ le programme « if $R_j = 0$ then $P$ else $P'$ ». La fonction $f_Q$ est donc donnée par

$$\begin{array}{rcll} f_Q(\langle x_0, \ldots, x_k\rangle) & = & f_P(\langle x_0, \ldots, x_k\rangle) & \text{si } x_j = 0 \\ & = & f_{P'}(\langle x_0, \ldots, x_k\rangle) & \text{sinon.} \end{array}$$

Supposons à présent que la proposition est vraie pour des programmes $P$, via des fonctions $f_P$, et soit $Q$ de forme suivante : « for $i = 1$ to $R_j$ do $P$ ». La fonction $f_Q$ est donnée par :

$$f_Q(\langle x_0, \ldots, x_k\rangle) = g(\langle x_0, \ldots, x_k\rangle, x_j)$$

où

$$\begin{array}{rcl} g(\langle x_0, \ldots, x_k\rangle, 0) & = & \langle x_0, \ldots, x_k\rangle \\ g(\langle x_0, \ldots, x_k\rangle, z+1) & = & f_P\big(g(\langle x_0, \ldots, x_k\rangle, z)\big). \end{array}$$

Supposons à présent la proposition vraie pour des programmes $P, P'$, via des fonctions $f_P, f_{P'}$, et soit $Q$ composé des instructions de $P$ suivies de celles de $P'$. Alors, $f_Q = f_{P'}\big(f_P(\langle x_0, \ldots, x_k\rangle)\big)$.

En utilisant chacun des cas décrits, on montre par récurrence que l'état des registres de tout programme structuré sans boucles `while` est une fonction primitive récursive. Il suffit alors de récupérer la valeur du registre $R_0$. ∎

### 3.4. Étude de la fonction d'Ackermann

Il existe plusieurs manières de voir que les fonctions calculables ne sont pas toutes primitives récursives. La suivante est la plus naturelle pour l'expert en calculabilité pour lequel les diagonalisations effectives n'ont plus de secret.

**Exercice 3.23. (⋆)** Montrer qu'il existe un ensemble calculable $A \subseteq \mathbb{N}$ tel que $n \mapsto \Phi_e(n)$ est une fonction primitive récursive pour tout $e \in A$ et tel que toute fonction primitive récursive a un code dans $A$. En déduire qu'il existe une fonction totale calculable qui n'est pas primitive récursive. ◇

Un exemple couramment donné de fonction non primitive récursive calculable est la fonction d'Ackermann.

**Définition 3.24 (Fonction d'Ackermann [3]).** On définit les diverses fonctions $A_n : \mathbb{N} \to \mathbb{N}$ par récurrence sur $n \in \mathbb{N}$ de la manière suivante :

- ▷ $A_0$ est la fonction $x \mapsto 2^x$ ;
- ▷ $A_{n+1}(x)$ est l'application $x$ fois de la fonction $A_n$ sur 1 :

$$A_n(A_n(\ldots(A_n(1)))).$$

Formellement,

$$\begin{aligned} A_0(x) &= 2^x \\ A_{n+1}(0) &= 1 \\ A_{n+1}(x) &= A_n(A_{n+1}(x-1)). \end{aligned}$$

La fonction d'Ackermann est la fonction $n \mapsto A_n(n)$. ♢

La fonction d'Ackermann a une croissance extrêmement rapide :

$$A(0) = 1, \qquad A(1) = 2, \quad A(2) = 16,$$

et $A(3)$ est déjà égal à 65 536 itérations de la fonction $x \mapsto 2^x$ (en commençant sur 0), c'est-à-dire :

$$A(3) = 2^{\left(2^{\left(\ldots^{(2^0)}\right)}\right)}, \text{ où la puissance est itérée 65 536 fois.}$$

Malgré sa très forte croissance, la fonction d'Ackermann est calculable : pour calculer $A_n(n)$, on peut utiliser une pile contenant soit des fonc-

tions $A_n$ (en pratique une représentation de ces fonctions), soit des entiers. Par exemple, si l'on empile $A_n$, $A_{n+1}$ et ensuite $k$, cela correspond au calcul $A_n(A_{n+1}(k))$. Ainsi le sommet de la pile est-il toujours un entier, et l'élément qui suit (s'il existe) est toujours une fonction. Aussi, pour calculer $A_n(n)$, on procède comme suit.

1. On empile $A_n$, puis on empile $n$.
2. Tant que la pile contient plus d'un élément :
   (a) on dépile l'entier $k$, puis on dépile la fonction $A_m$ ;
   (b) si $m = 0$, on empile $2^k$ ;
   (c) sinon, si $k = 0$ on empile $1$ ;
   (d) sinon, on empile $A_{m-1}$, puis $A_m$ et enfin $k - 1$.

L'algorithme s'arrêtera quand la pile ne contiendra plus qu'un élément, le résultat du calcul de $A_n(n)$.

Nous laissons en exercice la preuve que la fonction d'Ackermann croît plus vite que toute fonction primitive récursive, et n'est donc pas elle-même primitive récursive.

**Exercice 3.25 (Cori et Lascar[42]). ($\star$)**

(1) Montrer que $A_n(x) > x$ pour tous $n, x$.

(2) Montrer que la fonction $A_n$ est strictement croissante pour tout $n$.

(3) Montrer que la fonction $n \mapsto A_n(x)$ est croissante pour tout $x$.

(4) Montrer que $A_{n+1}(x + y) \geqslant A_n^{(y)}(x)$ pour tous $n, x, y$, où $f^{(m)}(x)$ dénote $f(f(\dots f(x)\dots))$ où l'application de $f$ est itérée $m$ fois.

(5) Montrer que pour tout $n$, on a $A_{n+2}(x) > A_n^{(x+1)}(x + 1)$ pour presque tout $x$.

(6) Soit $n > 0$, et soit $f : \mathbb{N}^n \to \mathbb{N}$. On note $P(f)$ le prédicat

$$\exists k\ \forall^\infty x_1, \dots, x_n\ A_k(\max(x_1, \dots, x_n)) > f(x_1, \dots, x_n).$$

Montrer que $P(f)$ est vrai pour toute fonction primitive récursive $f$.

En déduire que la fonction d'Ackermann n'est pas primitive récursive. $\diamond$

Les boucles de type `while` sont donc indispensables pour calculer certaines fonctions, et avec elles, surviennent la possibilité d'écrire des programmes qui ne s'arrêtent pas.

### 3.5. Les fonctions calculables sont récursives

Nous nous apprêtons à présent à montrer que le schéma de minimisation des fonctions récursives permet de calculer n'importe quelle fonction programmable avec ou sans boucles `while`. Pour ce faire, nous commençons par voir comment simuler des structures de listes de taille arbitraire par des entiers. Cela semble en effet pour le moment être un manque de notre modèle de machine à registres et de programmes structurés. Nous avons par exemple besoin d'une pile pour calculer la fonction d'Ackermann via l'algorithme mentionné ci-dessus.

**Proposition 3.26.** Il existe une bijection $[\ ] : \bigcup_{n\in\mathbb{N}} \mathbb{N}^n \to \mathbb{N}$ (où l'on note $[x_1,\dots,x_n]$ l'entier correspondant au $n$-uplet $(x_1,\dots,x_n)$), telle que les opérations suivantes sont primitives récursives.

1. La fonction :: d'ajout en tête de liste définie par
$$a :: [x_1,\dots,x_n] = [a,x_1,\dots,x_n].$$
2. Les fonctions hd et tl de tête et queue de liste, définies par
$$\begin{array}{rclcrcl} \mathrm{hd}([\ ]) & = & 0 & & \mathrm{tl}([\ ]) & = & 0 \\ \mathrm{hd}(x :: l) & = & x & & \mathrm{tl}(x :: l) & = & l. \end{array}$$
3. La fonction $|\ |$ de taille d'une liste définie par $|[x_1,\dots,x_n]| = n$.
4. Les fonctions $\mathrm{get}, \mathrm{set}$ telles que
$$\begin{array}{rcll} \mathrm{get}([x_0,\dots,x_{n-1}],i) & = & x_i & \text{si } i<n \\ \mathrm{get}([x_0,\dots,x_{n-1}],i) & = & 0 & \text{sinon} \end{array}$$
et
$$\begin{array}{rcll} \mathrm{set}([x_0,\dots,x_{n-1}],a,i) & = & [x_0,\dots,x_{i-1},a,x_{i+1},\dots,x_{n-1}] & \text{si } i<n \\ \mathrm{set}([x_0,\dots,x_{n-1}],a,i) & = & [x_0,\dots,x_{n-1}] & \text{sinon.} \end{array}$$

⋆

Preuve. La bijection $[\ ] : \bigcup_{n1\mathbb{N}} \mathbb{N}^n$ est définie par récurrence en commençant par la liste vide $[\ ] = 0$ et en appliquant l'opération d'ajout en tête de liste définie par $a :: l = 1 + \alpha_2(a,l)$, où $\alpha_2 : \mathbb{N}^2 \to \mathbb{N}$ est la bijection de l'exercice 2-3.7. Il est clair que la fonction d'ajout en tête est primitive récursive, de même que les fonctions hd et tl obtenues grâce aux fonctions inverses de $(x_1,x_2) \mapsto \langle x_1,x_2\rangle$. Montrons que le codage des listes obtenu ainsi est bien une bijection.

Montrons par récurrence que deux listes de tailles différentes ne peuvent pas être codées par le même élément. Le code de la liste vide est 0 et le code d'une liste non vide est de la forme $1 + \alpha_2(a,l) \neq 0$. Donc, le code de la liste vide est toujours différent du code d'une liste non vide.

Supposons à présent que toutes les listes de taille $n$ ont un code différent de celui des listes de taille $m > n$. Montrons que toutes les listes de taille $n+1$ ont un code différent de celui des listes de taille $m > n+1$. Les codes des listes de taille $n+1$ sont de la forme $1 + \alpha_2(a, l_1)$ pour $l_1$ le code d'une liste de taille $n$. Les codes des listes de taille $m > n+1$ sont de la forme $1 + \alpha_2(b, l_2)$ pour $l_2$ le code d'une liste de taille $m > n$. Par hypothèse de récurrence, on a forcément $l_1 \neq l_2$ et comme $\alpha_2$ est injectif, on a forcément $1 + \alpha_2(a, l_1) \neq 1 + \alpha_2(b, l_2)$. Donc, les codes des listes de taille $n+1$ sont différents des codes des listes de taille $m > n+1$. Par récurrence, on en déduit que les codes de listes de tailles différentes sont différents.

Montrons à présent par récurrence sur $k$ que si $(a_1, \ldots, a_k) \neq (b_1, \ldots, b_k)$, on a alors $[a_1, \ldots, a_k] \neq [b_1, \ldots, b_k]$.
Pour $k = 1$, on a $a_1 \neq b_1$ implique $1 + \alpha_2(a_1, 0) \neq 1 + \alpha_2(b_1, 0)$, car $\alpha_2$ est injective. On a donc $[a_1] \neq [b_1]$. Supposons que ce soit le cas pour $k$, et montrons que c'est le cas pour $k+1$. Supposons $(a_1, \ldots, a_{k+1}) \neq (b_1, \ldots, b_{k+1})$. Si $a_1 \neq b_1$, on a alors $1 + \alpha_2(a_1, [a_2, \ldots, a_{k+1}]) \neq 1 + \alpha_2(b_1, [b_2, \ldots, b_{k+1}])$, car $\alpha_2$ est injective. Si $(a_2, \ldots, a_{k+1}) \neq (b_2, \ldots, b_{k+1})$, alors par hypothèse de récurrence on a $[a_2, \ldots, a_{k+1}] \neq [b_2, \ldots, b_{k+1}]$, et donc

$$1 + \alpha_2(a_1, [a_2, \ldots, a_{k+1}]) \neq 1 + \alpha_2(b_1, [b_2, \ldots, b_{k+1}]),$$

car $\alpha_2$ est injective. Par récurrence, pour tout $k$ on a donc

$$(a_1, \ldots, a_k) \neq (b_1, \ldots, b_k) \text{ implique } [a_1, \ldots, a_k] \neq [b_1, \ldots, b_k].$$

La fonction $[\,]$ est ainsi injective.

Montrons à présent que $[\,]$ est surjective. Supposons par l'absurde que ce n'est pas le cas. Dans ce cas, il existe un plus petit $n$ tel que $n$ n'est le code d'aucune liste. Notons que l'on a forcément $n > 0$, car 0 est le code de la liste vide. Aussi, comme $\alpha_2$ est surjective, il existe (a,b) tel que $\alpha_2(a, b) = n - 1$, et donc tel que $1 + \alpha_2(a, b) = n$. On a par ailleurs nécessairement $b \leqslant n - 1 < n$. Aussi, par minimalité de $n$, il doit exister une liste $l$ dont $b$ est le code. Donc, $n$ est le code de la liste $a :: l$, ce qui contredit notre hypothèse sur $n$.

Afin de montrer que les fonctions $|\,|$, get et set sont primitives récursives, on donne une définition primitive récursive de la fonction $\mathrm{tl}(l, n)$ qui ampute une liste $l$ de ses $n$ premiers éléments :

$$\begin{aligned} \mathrm{tl}(l, 0) &= l \\ \mathrm{tl}(l, n+1) &= \mathrm{tl}(\mathrm{tl}(l, n)). \end{aligned}$$

La taille est alors définie via le schéma de minimisation bornée par l'entier $|l| = \min\{n \leqslant l \,:\, \mathrm{tl}(l, n) = [\,]\}$. La fonction get a la définition primitive récursive suivante : $\mathrm{get}(l, n) = \mathrm{hd}(\mathrm{tl}(l, n))$. La fonction set est quant à elle

définie en deux fois, en ajoutant d'abord un paramètre supplémentaire :

$$\begin{array}{rcl} \mathrm{set}(l,a,i) & = & \mathrm{set}(l,a,i,i) \\ \mathrm{set}(l,a,n,0) & = & a :: \mathrm{tl}(l,\mathrm{succ}(n)) \\ \mathrm{set}(l,a,n,i+1) & = & \mathrm{get}(l,n-\mathrm{succ}(i)) :: \mathrm{set}(l,a,n,i) \end{array}$$ ∎

Nous avons à présent tous les éléments nécessaires pour montrer que les fonctions calculables par des programmes structurés sont récursives.

**Théorème 3.27**
*Toute fonction calculable par un programme goto (et donc aussi par un programme structuré) est une fonction générale récursive.*

Le reste de la section est consacré à la preuve, pour laquelle nous avons besoin de fixer un codage des programmes goto et des machines à registre.

**Codage des programmes goto**. On code les instructions des programmes goto comme suit :

- ▷ $\ll R_i = R_i + 1 \gg$ est codé par $\langle 0, i\rangle$
- ▷ $\ll R_i = R_i - 1 \gg$ est codé par $\langle 1, i\rangle$
- ▷ $\ll R_i = 0 \gg$ est codé par $\langle 2, i\rangle$
- ▷ $\ll$ if $R_i = 0$ goto $n_1$ else $n_2 \gg$ est codé par $\langle 3, \langle i, \langle n_1, n_2\rangle\rangle\rangle$
- ▷ $\ll$ goto $n \gg$ est codé par $\langle 4, n\rangle$.

Pour des raisons d'uniformité, il sera utile d'avoir une borne sur l'indice maximal des registres utilisés. Le code d'un programme goto est simplement donné par le code : $\langle k, [I_1, \ldots, I_n]\rangle$, où $k$ est tel que le programme utilise au plus les registres $R_0, \ldots, R_k$ et où $I_e$ est le code de la $e$-ième instruction pour $1 \leqslant e \leqslant n$.

**Codage des machines à registres**. On fixe à présent un codage de l'état d'une machine à $k$ registres. Cet état est donné par la valeur des registres ainsi que par l'indice de la prochaine instruction à exécuter. Pour un nombre de registres $k$ donné, ce code est $\langle m, [R_0, \ldots, R_k]\rangle$, où $m$ est le numéro d'instruction et $R_i$ est la valeur du registre numéro $i$ pour $0 \leqslant i \leqslant k$.

**Fonction d'initialisation**. Fixons maintenant une fonction d'initialisation init, qui étant donné un code $e = \langle k, I\rangle$ d'un programme (où $I$ est une liste d'instructions), des valeurs $x_1, \ldots, x_n$ (pour $n \leqslant k$), donne le code représentant l'état de la machine à $k$ registres, au début du calcul.

$$\mathrm{init}(\langle k, I\rangle, x_1, \ldots, x_n) \quad = \quad \langle 0, 0 :: x_1 :: \ldots :: x_n :: \mathrm{aux}(k-n)\rangle$$

avec aux définie par

$$\begin{array}{rcl} \mathrm{aux}(0) & = & [\,] \\ \mathrm{aux}(k+1) & = & 0 :: \mathrm{aux}(k). \end{array}$$

Il est clair que la fonction init est primitive récursive.

**Fonctions de transition 1**. On définit une fonction de transition $\mathrm{tr}_1$, qui étant donné le code $e = \langle k, I\rangle$ d'un programme et le code $p = \langle m, R\rangle$ de l'état d'une machine, renvoie le numéro de la prochaine instruction à exécuter à l'étape de calcul suivante. On va utiliser pour cela une fonction $\mathrm{cur} : \mathbb{N} \to \mathbb{N}$ qui, étant donné le code $e = \langle k, I\rangle$ d'un programme et le code $p = \langle m, R\rangle$ de l'état d'une machine, permet d'obtenir l'instruction courante de la machine :

$$\mathrm{cur}(\langle k, I\rangle, \langle m, R\rangle) = \mathrm{get}(I, m).$$

En utilisant la fonction cur, les fonctions $\pi_1, \pi_2$ telles que $n = \langle \pi_1(n), \pi_2(n)\rangle$ on définit $\mathrm{tr}_1(e, p)$ comme étant :

$$\begin{array}{lll}
\pi_1(p) + 1 & \text{si} & \pi_1(\mathrm{cur}(e,p)) \leqslant 2 \\
\pi_1(\pi_2(\pi_2(\mathrm{cur}(e,p)))) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 3 \text{ et} \\
 & & \mathrm{get}(\pi_2(p), \pi_1(\pi_2(\mathrm{cur}(e,p)))) = 0 \\
\pi_2(\pi_2(\pi_2(\mathrm{cur}(e,p)))) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 3 \text{ et} \\
 & & \mathrm{get}(\pi_2(p), \pi_1(\pi_2(\mathrm{cur}(e,p)))) \neq 0 \\
\pi_2(\mathrm{cur}(e,p)) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 4.
\end{array}$$

Il est clair que $\mathrm{tr}_1$ est primitive récursive.

**Fonctions de transition 2**. On définit à présent une fonction de transition $\mathrm{tr}_2$ qui étant donné le code $e$ d'un programme et le code $p$ de l'état d'une machine, permet d'obtenir l'état des registres de la machine à l'étape de calcul suivante.

Pour cela l'on utilisera deux fonctions primitives récursives $\mathrm{inc} : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ et $\mathrm{dec} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, telles que

$$\begin{array}{rcl}
\mathrm{inc}([x_0, \ldots, x_n], i) & = & [x_0, \ldots, x_i + 1, \ldots, x_n] \\
\mathrm{dec}([x_0, \ldots, x_n], i) & = & [x_0, \ldots, \max(0, x_i - 1), \ldots, x_n]
\end{array}$$

définies de la manière suivante.

$$\begin{array}{rcl}
\mathrm{inc}(l, i) & = & \mathrm{set}(l, \mathrm{succ}(\mathrm{get}(l, i)), i) \\
\mathrm{dec}(l, i) & = & \mathrm{set}(l, \mathrm{pred}(\mathrm{get}(l, i)), i).
\end{array}$$

On peut à présent définir $\mathrm{tr}_2(e, p)$ comme étant :

$$\begin{array}{rll}
\mathrm{inc}(\pi_2(p), \pi_2(\mathrm{cur}(e,p))) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 0 \\
\mathrm{dec}(\pi_2(p), \pi_2(\mathrm{cur}(e,p))) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 1 \\
\mathrm{set}(\pi_2(p), 0, \mathrm{cur}(e,p)) & \text{si} & \pi_1(\mathrm{cur}(e,p)) = 2 \\
\pi_2(p) & \text{sinon.} &
\end{array}$$

Il est clair que $\mathrm{tr}_2$ est primitive récursive.

**Fin de la preuve**. On définit à présent la fonction $\mathrm{tr} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ de transition d'un état à un autre :

$$\mathrm{tr}(e,p) \quad = \quad \langle \mathrm{tr}_1(e,p), \mathrm{tr}_2(e,p) \rangle.$$

On définit ensuite la fonction primitive récursive

$$\mathrm{st} : \mathbb{N} \times \mathbb{N}^n \times \mathbb{N} \to \mathbb{N}$$

telle que $\mathrm{st}(e, x_1, \ldots, x_n, t)$ renvoie l'état de la machine qui exécute le programme $e$, avec les registres $R_1, \ldots, R_n$, initialisés à $x_1, \ldots, x_n$ respectivement, après $t$ étapes de calcul.

$$\begin{aligned} \mathrm{st}(e, x_1, \ldots, x_n, 0) &= \mathrm{init}(e, x_1, \ldots, x_n) \\ \mathrm{st}(e, x_1, \ldots, x_n, t+1) &= \mathrm{tr}(e, \mathrm{st}(e, x_1, \ldots, x_n, t)). \end{aligned}$$

On arrive enfin à l'étape pour laquelle on a besoin de schéma de minimisation, laissant la possibilité à une fonction de ne pas être définie sur certaines entrées. La fonction récursive $\mathrm{time} : \mathbb{N} \times \mathbb{N}^n \to \mathbb{N}$ donne le plus petit temps de calcul nécessaire pour que la machine s'arrête, c'est-à-dire arrive à un numéro d'instruction plus grand que le nombre d'instructions du programme. La fonction sera définie si, et seulement si, la machine s'arrête pour le programme et les entrées correspondantes.

$$\mathrm{time}(e, x_1, \ldots, x_n) \quad = \quad \min\{t \in \mathbb{N} \ : \ \pi_1(\mathrm{st}(e, x_1, \ldots, x_n, t)) \geqslant |\pi_2(e)|\}.$$

Finalement, voici la fonction récursive qui correspond au calcul de la machine de code $e$. On lance la fonction de transition pour le nombre d'étapes nécessaires avant que la machine ne s'arrête, et l'on renvoie la valeur du registre $R_0$ :

$$f(x_1, \ldots, x_n) \quad = \quad \mathrm{hd}(\pi_2(\mathrm{st}(e, x_1, \ldots, x_n, \mathrm{time}(e, x_1, \ldots, x_n)))).$$

Cela conclut la démonstration.

### 3.6. Conséquences

D'après la preuve précédente, étant donné notre codage d'un programme par un entier $e$, son exécution pour $t$ étapes de calcul est une fonction primitive récursive et donc elle-même calculable par un programme structuré. La recherche de ce plus petit temps de calcul peut se faire à l'aide d'une boucle `while`. Remarquons de plus que la preuve est uniforme : la même fonction primitive récursive s'adapte en fonction de tout code $e$ d'un programme.

Cela permet d'obtenir le théorème 3-3.1 de l'existence d'un programme universel, utilisé tout au long du livre, via la notation $\Phi_e$ pour le programme de code $e$.

**Théorème (3-3.1)**
*Soit $n \in \mathbb{N}^*$. Il existe un code $e$ de programme informatique pour lequel $\Phi_e : \mathbb{N}^{n+1} \to \mathbb{N}$ est tel que, pour tous $x_1, \ldots, x_n$, on a*

- ▷ $\Phi_e(a, x_1, \ldots, x_n)\uparrow$ *ssi* $\Phi_a(x_1, \ldots, x_n)\uparrow$ ;
- ▷ $\Phi_e(a, x_1, \ldots, x_n)\downarrow = y$ *ssi* $\Phi_a(x_1, \ldots, x_n)\downarrow = y$.

Le code $e$ du théorème ci-dessus est un code de la fonction

$$(x_1, \ldots, x_n) \mapsto \mathrm{hd}(\pi_2(\mathrm{st}(a, x_1, \ldots, x_n, \mathrm{time}(a, x_1, \ldots, x_n))))$$

donnée à la fin de la section précédente. La fonction

$$(a, x_1, \ldots, x_n) \mapsto \mathrm{time}(a, x_1, \ldots, x_n)$$

qui cherche le plus petit temps de calcul tel que le programme s'arrête est la seule qui utilise le schéma de minimisation. Notons que cela permet aussi de donner une définition mathématique précise aux notations

$$\Phi_a(x_1, \ldots, x_n)[t]\downarrow \quad \text{et} \quad \Phi_a(x_1, \ldots, x_n)[t]\uparrow .$$

Elles correspondent respectivement aux prédicats primitifs récursifs :

$$\begin{aligned} & \exists s \leqslant t \ \pi_1(\mathrm{st}(e, x_1, \ldots, x_n, s)) \geqslant |\pi_2(e)| \\ \text{et} \quad & \forall s \leqslant t \ \pi_1(\mathrm{st}(e, x_1, \ldots, x_n, s)) < |\pi_2(e)|. \end{aligned}$$

Chapitre 7

# Immunité et croissance de fonction

La calculabilité étudie la puissance calculatoire des ensembles d'entiers, modulo la réduction Turing. Dans ce chapitre, nous allons étudier en particulier deux grandes familles de propriétés calculatoires, à savoir la capacité à calculer des ensembles difficiles à décrire (ensemble immune, hyperimmune, effectivement immune), et la capacité à calculer des fonctions à croissance rapide (fonction hyperimmune, fonction dominante). Prenons quelques exemples.

**Exemple 1.** D'après l'exercice 3-7.10, tout ensemble infini c. e. contient un sous-ensemble infini calculable. Quelle puissance calculatoire faut-il pour obtenir un ensemble infini ne possédant pas de sous-ensemble infini calculable ? Nous étudierons cela dans la section 1 sous le concept d'ensemble immune.

**Exemple 2.** D'après le théorème 3-6.2 du point fixe de Kleene, pour toute fonction totale calculable $f : \mathbb{N} \to \mathbb{N}$, il existe un code $e$ tel que $\Phi_{f(e)} = \Phi_e$. Quelle est la puissance calculatoire d'une fonction sans point fixe ? Cette notion sera étudiée dans la section 2.

**Exemple 3.** Toute fonction calculable est trivialement dominée par une fonction calculable. Quelle puissance calculatoire faut-il pour calculer une fonction qui n'est dominée par aucune fonction calculable ? Ce sera le sujet de la section 4.

Il est difficile de se faire une intuition sur la puissance calculatoire *a priori* de propriétés formulées de manière aussi diverses, et en particulier de les comparer. Les propriétés tirées des trois exemples précédents admettent cependant des caractérisations qui rendront cette comparaison plus aisée. De manière générale, l'existence de nombreuses caractérisations d'une même puissance calculatoire avec des formulations très diverses est un gage de robustesse de la notion. C'est en particulier le cas des propriétés étudiées dans ce chapitre.

# 1. Ensembles immunes

La première famille de propriétés calculatoires sur les ensembles relève de la capacité à approximer les éléments d'un ensemble. Un ensemble est calculable s'il est possible de déterminer calculatoirement quels éléments lui appartiennent ou non. Au niveau suivant, un ensemble est calculatoirement énumérable s'il existe une procédure calculable pour lister tous les éléments qui lui appartiennent, mais potentiellement dans le désordre, ce qui fait qu'il n'est généralement pas possible d'être certain qu'un élément n'appartient pas à l'ensemble. Nous allons maintenant étudier la puissance calculatoire des ensembles infinis pour lesquels il n'est même pas possible d'énumérer de manière calculable une quantité infinie de ses éléments.

**Définition 1.1.** Un ensemble infini $A \subseteq \mathbb{N}$ est *immune* s'il ne contient pas de sous-ensemble infini c. e. ♢

Comme nous l'avons vu, tout ensemble infini c. e. contient un sous-ensemble infini calculable. Ainsi, de manière équivalente, un ensemble infini est immune si, et seulement si, il ne contient pas de sous-ensemble infini calculable. En particulier, tout ensemble immune $A$ est nécessairement non calculable, car $A$ serait alors son propre sous-ensemble infini calculable contredisant son immunité.

L'immunité est une notion d'ensemble, mais non pas de degré. En effet, si $A$ est un ensemble immune, l'ensemble $A \oplus \mathbb{N} = \{2n : n \in A\} \sqcup \{2n+1 : n \in \mathbb{N}\}$ est de même degré Turing que $A$, mais $A \oplus \mathbb{N}$ possède le sous-ensemble infini calculable $\{2n+1 : n \in \mathbb{N}\}$. Inversement, tout degré Turing non calculable contient un ensemble immune, comme le montre la proposition suivante.

**Proposition 1.2.** Tout ensemble non calculable est Turing équivalent à un ensemble immune. ⋆

Preuve. Soit $A$ un ensemble non calculable, et soit $B = \{\sigma \in 2^{<\mathbb{N}} : \sigma \prec A\}$ l'ensemble des segments initiaux de $A$. En particulier, $A \equiv_T B$. Il est par ailleurs évident que tout sous-ensemble infini de $B$ permet de calculer des

segments initiaux arbitrairement grands de $A$, et donc $A$ (ainsi d'ailleurs que $B$). Comme $A$ n'est pas calculable, $B$ ne possède pas de sous-ensemble infini calculable. ∎

La notion d'ensemble immune possède deux renforcements orthogonaux, à savoir les ensembles effectivement immunes, et les ensembles hyperimmunes. Ces deux notions sont des propriétés calculatoires fondamentales en calculabilité, et nous verrons pour chacune d'entre elles plusieurs définitions équivalentes.
Rappelons que $W_e$ désigne l'ensemble c. e. de code $e$ : $\{n \ : \ \Phi_e(n)\downarrow\}$.

**Définition 1.3.** Un ensemble infini $A \subseteq \mathbb{N}$ est *effectivement immune* s'il existe une fonction totale calculable $h : \mathbb{N} \to \mathbb{N}$ telle que pour tout code $e$, si $|W_e| \geqslant h(e)$ alors $W_e \not\subseteq A$. ♢

Intuitivement, un ensemble infini $A$ est effectivement immune si non seulement les ensembles infinis finissent par se tromper et énumérer un élément en-dehors de $A$, mais plus encore, cette erreur doit arriver après suffisamment peu d'éléments énumérés, en fonction du code de l'énumération. En particulier, tout ensemble effectivement immune est immune.

Nous verrons que le concept d'immunité effective est particulièrement digne d'intérêt du point de vue des degrés Turing. La puissance de calcul correspondant à la capacité de calculer un ensemble effectivement immune possède de nombreuses caractérisations qui seront étudiées dans la section 2.

Rappelons que le *code canonique* d'un ensemble fini $F$ est l'entier

$$n = \sum_{i \in F} 2^i.$$

Soit $D_0, D_1, \ldots$ la collection des ensembles finis telle que $D_n$ est canoniquement codé par $n$, pour tout $n \in \mathbb{N}$.

**Définition 1.4.** Un *tableau* est une collection d'ensembles finis mutuellement disjoints $F_0, F_1, \ldots$ Un tableau $F_0, F_1, \ldots$ est *c. e.* s'il existe une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ telle que pour tout $n$, $F_n = D_{f(n)}$. Un ensemble infini $A$ est *hyperimmune* si pour tout tableau c. e. $F_0, F_1, \ldots$, il existe un entier $n$ tel que $F_n \cap A = \emptyset$. ♢

Cette définition plus complexe, formalise l'idée selon laquelle non seulement il n'est pas possible de lister calculatoirement une infinité d'éléments de $A$, mais plus encore il n'est même pas possible de lister une infinité de « blocs » finis d'éléments disjoints deux à deux, tels que chaque bloc contient au moins un élément de $A$.

Tout comme pour le concept d'immunité effective, c'est l'extension de l'hyperimmunité aux degrés Turing qui nous intéressera surtout dans la suite. La puissance de calcul correspondant à la capacité de calculer un ensemble hyperimmune possède de nombreuses caractérisations qui seront étudiées dans la section 4.

**Exercice 1.5.** Montrer que tout ensemble hyperimmune est immune. ◇

## 2. Fonctions DNC

Nous voyons à présent un exemple de degré Turing remarquable, dont l'étude remonte sans doute aux travaux d'Arslanov [8], qui étudia les degrés Turing permettant d'échapper au fameux théorème du point fixe de Kleene : étant donné une fonction calculable $f$, il existe $e$ tel que $\Phi_e = \Phi_{f(e)}$. Quelle puissance est nécessaire pour calculer une fonction $f$ pour laquelle ce n'est pas le cas ? Ces travaux ont été étendus par Jockusch, Lerman, Soare, et Solovay qui ont trouvé une caractérisation équivalente, qui constitue aujourd'hui la définition moderne de degré DNC.

**Définition 2.1.** Une fonction $f : \mathbb{N} \to \mathbb{N}$ est *diagonalement non calculable* (DNC) si $f(n) \neq \Phi_n(n)$ pour tout $n$. ♢

Insistons sur le fait que la fonction $f$ doit être totale dans la définition ci-dessus. Notons que si $\Phi_n(n)\uparrow$, il n'y a pas de restriction sur la valeur de $f(n)$. Un degré Turing est DNC s'il contient une fonction DNC.

**Exercice 2.2. (⋆)** Montrer que les degrés DNC sont clos par le haut, c'est-à-dire que si un ensemble calcule une fonction DNC, son degré est lui-même DNC. ◇

La notion de degré DNC présente de nombreuses applications dans les liens entre calculabilité et aléatoire algorithmique, tout comme en mathématiques à rebours. Nous verrons en particulier avec le corollaire 18-4.3 que les DNC sont nombreux du point de vue de la mesure, mais d'après la proposition 10-3.36 peu nombreux du point de vue des catégories de Baire (nous verrons en particulier l'existence de degrés non DNC et non calculables).

Observons tout d'abord qu'il n'existe pas de fonction DNC calculable, par un simple argument diagonal, ce qui est à l'origine de l'appellation « diagonalement non calculable ». La proposition suivante montre en revanche que l'on peut calculer une fonction DNC à l'aide du problème de l'arrêt.

**Proposition 2.3.** Le degré Turing $\mathbf{0}'$ du problème de l'arrêt est un degré DNC. ⋆

PREUVE. Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction $\emptyset'$-calculable, qui pour une entrée $e$, renvoie $1 - \Phi_e(e)$ si $e \in \emptyset'$, et 0 sinon. Cette fonction est DNC, car elle est totale, et lorsque $\Phi_e(e)\downarrow$, elle renvoie une valeur différente. Comme $\emptyset'$ calcule une fonction DNC, et que les degrés DNC sont clos par le haut, $\mathbf{0}'$ est DNC. ∎

Nous verrons dans la section 3 que $\mathbf{0}'$ est le seul degré à la fois DNC et c. e. Il est clair que les degrés DNC sont en quantité indénombrable, car c'est également le cas des degrés au-dessus de $\mathbf{0}'$. Nous verrons avec la proposition 10-3.36 que les degrés non DNC sont eux aussi en quantité indénombrable.

Pour le moment, nous nous attachons à montrer que la notion de degré DNC est naturelle, dans le sens où elle possède de nombreuses caractérisations via des formulations très différentes (et nous en verrons d'autres encore dans la partie II sur l'aléatoire algorithmique).

**Définition 2.4.** Une fonction $f$ est *libre de point fixe* si $\Phi_n \neq \Phi_{f(n)}$ pour tout $n$. ♢

**Théorème 2.5 (Jockusch, Lerman, Soare, et Solovay [101])**
*Soit $X \in 2^{\mathbb{N}}$. Alors, les énoncés suivants sont équivalents.*

*(1) $X$ calcule une fonction diagonalement non calculable.*

*(2) $X$ calcule une fonction libre de point fixe.*

PREUVE. Montrons (1) ⇒ (2). Soit $g \leqslant_T X$ telle que $g(n) \neq \Phi_n(n)$ pour tout $n$. Alors également, $f \leqslant_T X$ telle que $f(n)$ est un code pour une fonction calculable définie uniquement sur l'entrée $n$, et qui à $n$ associe $g(n)$. On a en particulier $\Phi_{f(n)}(n)\downarrow = g(n)$. Supposons qu'il existe $n$ tel que $\Phi_{f(n)}(m) = \Phi_n(m)$ pour tout $m$. Alors, $\Phi_n(n)\downarrow = \Phi_{f(n)}(n)\downarrow = g(n)$, ce qui contredit la définition de $g$. Donc, $f$ est une fonction libre de point fixe.

Montrons (2) ⇒ (1). Soit $f \leqslant_T X$, une fonction libre de point fixe. Alors, à partir de $X$, on calcule la fonction $g : \mathbb{N} \to \mathbb{N}$ qui sur l'entier $n$ crée le code $e_n$ de la fonction $m \mapsto \Phi_{\Phi_n(n)}(m)$, et renvoie $f(e_n)$. On utilise ici le même abus de notation que dans la preuve du point fixe de Kleene : si $\Phi_n(n) \uparrow$, alors $m \mapsto \Phi_{\Phi_n(n)}(m)$ désigne la fonction nulle part définie. Notons que $g$ est totale, car elle ne cherche pas à faire le calcul $\Phi_n(n)$. Supposons que $g$ ne soit pas DNC, c'est-à-dire qu'il existe $n$ tel que $g(n) = \Phi_n(n)$. Par définition de $g$, $f(e_n) = \Phi_n(n)$. En particulier, $\Phi_{f(e_n)} = \Phi_{\Phi_n(n)} = \Phi_{e_n}$. La fonction $f$ n'est donc pas libre de point fixe, ce qui contredit la définition de $f$. Donc, $g$ est une fonction DNC. ∎

Voyons à présent l'équivalence entre degré DNC et degré effectivement immune. La troisième équivalence du théorème ci-dessous est plus technique, mais présente un grand intérêt et sera réutilisée par la suite. Il s'agit en fait d'un renforcement de la notion d'être DNC : considérons la suite $(A_n)_{n\in\mathbb{N}}$ d'ensembles c. e. définie par

$$A_n = \begin{cases} \{\Phi_n(n)\} & \text{si } \Phi_n(n)\downarrow \\ \emptyset & \text{sinon.} \end{cases}$$

Être de degré DNC est la capacité de calculer une fonction $f$ telle que pour tout $n$ entier, $f(n) \notin A_n$. La troisième équivalence étend ce résultat à l'énumération uniforme $(W_n)_{n\in\mathbb{N}}$ de tous les ensembles c. e. dont on sait borner le nombre d'éléments par un entier $n$.

**Théorème 2.6**
*Soit $X \in 2^{\mathbb{N}}$. Les énoncés suivants sont équivalents.*

*(1) $X$ calcule une fonction diagonalement non calculable.*

*(2) $X$ calcule un ensemble effectivement immune.*

*(3) $X$ calcule une fonction $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ telle que pour $e, n \in \mathbb{N}$,*

$$|W_e| \leqslant n \;\Rightarrow\; h(e,n) \notin W_e.$$

Preuve. (1) $\Rightarrow$ (3). Soit $f \leqslant_T X$ une fonction DNC. Nous décrivons un processus uniforme en $e, n$ afin de calculer une valeur qui, sous l'hypothèse $|W_e| \leqslant n$, n'est pas dans $W_e$. Pour chaque $0 \leqslant i < n$, on calcule le code $u(e,i)$ de la fonction partielle calculable qui, pour toute entrée, cherche le $i$-ième élément $k$ dans l'énumération de $W_e$, s'il existe. Si un tel élément $k$ est trouvé, la fonction l'interprète comme le $n$-uplet $\langle k_0, \ldots, k_{n-1}\rangle$, et renvoie $k_i$. Sinon, la fonction ne s'arrête pas. Notons que $\Phi_{u(e,i)}$ est soit une fonction constante, soit la fonction définie nulle part.

Montrons que la fonction $X$-calculable

$$h(e,n) = \langle f(u(e,0)), \ldots, f(u(e,n-1))\rangle$$

satisfait (3). Raisonnons par l'absurde, et supposons que $h(e,n) \in W_e$ avec $|W_e| \leqslant n$. Disons que $h(e,n)$ est le $i$-ième élément de $W_e$ dans l'ordre d'énumération. Alors, $\Phi_{u(e,i)}$ est la fonction constante qui trouve

$$k = h(e,n) = \langle f(u(e,0)), \ldots, f(u(e,n-1))\rangle$$

et renvoie le $i$-ième élément du tuple, c'est-à-dire $f(u(e,i))$. En particulier, $\Phi_{u(e,i)}(u(e,i)) \downarrow = f(u(e,i))$, ce qui contredit l'hypothèse selon laquelle $f$ est DNC.

(3) $\Rightarrow$ (2). Soit $h \leqslant_T X$ une fonction satisfaisant (3). Soit $D_0, D_1, \ldots$ une énumération effective de tous les ensembles finis tels que $n$ est le code

canonique de $D_n$. Soit $g : \mathbb{N} \to \mathbb{N}$ une fonction partielle calculable telle que si $|W_e| \geqslant e+1$ ; alors, $D_{g(e)} \subseteq W_e$ et $|D_{g(e)}| = e+1$. Nous allons définir une suite $X$-calculable infinie croissante d'entiers $x_0 < x_1 < \dots$ telle que pour tout $s$,

$$\forall e \leqslant s, \quad (|W_e| > e \;\Rightarrow\; D_{g(e)} \not\subseteq \{x_i : i \leqslant s\}). \tag{$\star$}$$

Il s'ensuit que $H = \{x_n : n \in \mathbb{N}\}$ est effectivement immune, car si $W_e \subseteq H$, alors $|W_e| \leqslant e$. Le fait que l'on veuille $x_i < x_{i+1}$ est simplement pour que l'ensemble $H$ soit $X$-calculable. Supposons maintenant que l'on ait déjà défini $x_0 < \dots < x_s$ satisfaisant ($\star$). Soit

$$W_{v(s)} = \{y : y \leqslant x_s\} \cup \bigcup_{e \leqslant s+1 \text{ t.q. } g(e)\downarrow} D_{g(e)}.$$

Le but est d'utiliser notre fonction $h$ pour trouver un élément qui n'est pas dans $W_{v(s)}$. Soit $x_{s+1}$ un tel élément — nous verrons après comment l'obtenir. Notons d'abord que $x_{s+1}$ est bien strictement plus grand que $x_s$. Notons ensuite que pour tout entier $e \leqslant s+1$ tel que $|W_e| > e$ et tel que $D_{g(e)} \not\subseteq \{x_i : i \leqslant s\}$, alors également $D_{g(e)} \not\subseteq \{x_i : i \leqslant s+1\}$. Notons finalement que pour $e \leqslant s$ tel que $|W_e| > e$, on a bien $D_{g(e)} \not\subseteq \{x_i : i \leqslant s\}$ par hypothèse, et pour $e = s+1$, si $|W_e| > e$, on a $D_{g(e)} \not\subseteq \{x_i : i \leqslant s\}$ nécessairement, car $D_{g(e)}$ possède alors $s+2$ éléments. Dans tous les cas, on aura bien la condition ($\star$) pour la suite $(x_i)_{i \leqslant s+1}$. Montrons à présent comment trouver $x_{s+1}$ à l'aide de la fonction $h$. L'ensemble $W_{v(s)}$ admet au plus $x_s + 1$ plus $1 + 2 + 3 + \dots + s + 2$ éléments, ce qui donne par la somme des termes d'une suite arithmétique $t_s = x_s + 1 + (s+2)(s+3)/2$. On définit alors $x_{s+1} = h(v(s), t_s)$. Par définition de $h$, $x_{s+1} \notin W_{v(s)}$. En particulier, $x_{s+1} > x_s$, et ($\star$) est satisfait pour $s+1$.

(2) $\Rightarrow$ (1) : Soit $H \leqslant_T X$ un ensemble effectivement immune. On considère $g : \mathbb{N} \to \mathbb{N}$ une fonction totale calculable telle que si $W_e \subseteq H$, alors $|W_e| < g(e)$. Nous allons montrer que $X$ calcule une fonction libre de point fixe. Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction $X$-calculable telle que $W_{f(e)}$ est l'ensemble des $g(e)$ premiers éléments de $H$. Notons que $f$ est $X$-calculable, mais que l'ensemble $W_{f(e)}$ est un ensemble c. e. qui, en particulier, n'a pas besoin de $X$ pour son énumération : on peut voir les éléments à énumérer comme étant « codés en dur » dans $f(e)$. Montrons que $f$ est libre de point fixe. Soit $e \in \mathbb{N}$ ; si $W_{f(e)} = W_e$, alors $W_e \subseteq H$, mais $|W_e| = |W_{f(e)}| = g(e)$, ce qui contredit le choix de $H$ et $g$. Comme $X$ calcule une fonction libre de point fixe, alors par le théorème 2.5, $X$ calcule une fonction DNC. ∎

Nous nous arrêterons là pour le moment. Nous verrons d'autres caractérisations de la notion de degrés DNC en rapport avec l'aléatoire algorithmique. Mentionnons pour finir une hiérarchie qui découle naturellement de la définition des degrés DNC.

**Définition 2.7.** Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction telle que

$$2 \leqslant f(n) \leqslant f(n+1).$$

Un ensemble $X \subseteq \mathbb{N}$ est de degré $\mathrm{DNC}_f$ si $X$ calcule une fonction $g$ telle que $g(n) < f(n)$ et $g(n) \neq \Phi_n(n)$ pour tout $n$. ♢

Un examen de la définition indique que plus la fonction $f$ croît lentement, plus il semble difficile pour un ensemble d'être $\mathrm{DNC}_f$. C'est effectivement le cas, comme on peut le voir avec le théorème suivant, dû à Ambos-Spies, Kjos-Hanssen, Lempp et Slaman [6].

**Théorème 2.8**
*Soit $f$ une fonction telle que $2 \leqslant f(n) \leqslant f(n+1)$. Il existe une fonction $g$ telle que $2 \leqslant g(n) \leqslant g(n+1)$ et avec $f < g$ pour laquelle $\mathrm{DNC_f} \subsetneq \mathrm{DNC_g}$, c'est-à-dire qu'il existe un ensemble $X$ qui calcule une fonction $\mathrm{DNC_g}$ mais qui ne calcule aucune fonction $\mathrm{DNC_f}$.*

Comme annoncé plus tôt, nous verrons de nombreux exemples de degrés non DNC et non calculables, mais il est informatif de s'assurer de leur existence par une construction directe.

**Exercice 2.9.** (⋆⋆) Montrer par la méthode des extensions finies qu'il existe des degrés $\Delta^0_2$ non DNC et non calculables. ◇

## 3. Critère de complétude d'Arslanov

Le critère de complétude d'Arslanov traduit d'une certaine manière une incompatibilité entre les degrés c. e. et DNC.

**Degré c. e.**

Un degré Turing est dit calculatoirement énumérable ou c. e. s'il contient un ensemble calculatoirement énumérable. Nous verrons dans le chapitre 13 que $\mathbf{0}'$ est loin d'être le seul degré c. e.

Il est aisé de calculer une fonction DNC à l'aide du problème de l'arrêt, comme le montre la proposition 2.3. En revanche, le critère de complétude d'Arslanov prouve que $\mathbf{0}'$ est le seul degré à la fois c. e. et DNC.

**Théorème 3.1 (Critère de complétude d'Arslanov [8])**
*Soit $A \in 2^{\mathbb{N}}$ un ensemble c. e. Alors, $A$ est Turing complet ssi $A$ calcule une fonction DNC.*

PREUVE. Par la proposition 2.3, si l'ensemble $A$ est Turing complet, il calcule une fonction DNC. Montrons la réciproque. Soit $(A_s)_{s\in\mathbb{N}}$ une approximation c. e. de $A$, c'est-à-dire avec $\lim_{s\to\infty} A_s = A$, et $A_s \subseteq A_{s+1}$. Notons que si $A_s \upharpoonright_n = A \upharpoonright_n$, alors aussi pour tout $t > s$ on aura $A_t \upharpoonright_n = A \upharpoonright_n$.

Soit $\Phi$ une fonctionnelle totale sur l'oracle $A$ et telle que $\Phi(A, n) \neq \Phi_n(n)$ pour tout $n$. Uniformément en $n$, on calcule le code $a_n$ de la fonction partielle qui sur toute entrée $k$ agit comme suit : cherche le plus petit $t$ tel que $\emptyset'[t](n) = 1$, puis si cela arrive et si $\Phi(A_t \upharpoonright_m, a_n)[t]$ s'arrête pour un certain $m \leqslant t$, alors renvoie la valeur de $\Phi(A_t \upharpoonright_m, a_n)[t]$, et sinon diverge. Notons que le code $a_n$ est défini en fonction de $a_n$ lui-même, ce qui est rendu possible grâce au théorème du point fixe.

Nous prétendons que pour tout $n$, si jamais $\emptyset'(n) = 1$, alors le plus petit $t$ tel que $\Phi(A_t \upharpoonright_m, a_n)[t]\downarrow$ pour $m \leqslant t$ tel que $A_t \upharpoonright_m = A \upharpoonright_m$, est strictement plus grand que le plus petit $t$ tel que $\emptyset'[t](n) = 1$. Supposons que ce ne soit pas le cas, c'est-à-dire il existe $n$ pour lequel $\emptyset'(n) = 1$ et pour lequel étant donné $t$ le plus petit entier tel que $\emptyset'[t](n) = 1$, on a également $\Phi(A_t \upharpoonright_m, a_n)[t]\downarrow$ pour $m \leqslant t$ tel que $A_t \upharpoonright_m = A \upharpoonright_m$. Dans ce cas, par la procédure décrite plus haut, on aura $\Phi_{a_n}(a_n) = \Phi(A_t \upharpoonright_m, a_n)$, et donc $\Phi_{a_n}(a_n) = \Phi(A \upharpoonright_m, a_n)$, ce qui contredit le fait que $\Phi$ calcule une fonction DNC sur l'oracle $A$.

L'oracle $A$ peut donc calculer $\emptyset'$ en cherchant simplement le plus petit temps de calcul $t$ tel que $A_t \upharpoonright_m = A \upharpoonright_m$ et $\Phi(A_t \upharpoonright_m, a_n)[t]\downarrow$ pour $m \leqslant t$, et puis en regardant si $\emptyset'[t](n) = 1$. Si c'est le cas, alors $\emptyset'(n) = 1$. Sinon $\emptyset'(n) = 0$. ∎

Le critère de complétude d'Arslanov connaît plusieurs extensions, notamment par Jockusch et al. [101]. Nous donnons l'une d'entre elles dans un exercice qui permettra de manipuler les principes de la preuve précédente.

**Exercice 3.2. (⋆⋆) (*Bienvenu et al.[16]*).** Une fonction $g$ est DNC relativement à $C$, noté DNC($C$), si $g(n) \neq \Phi_n(C, n)$ pour tout $n$. Soit $X \in 2^{\mathbb{N}}$ de degré DNC et soit $C$ un ensemble c. e. Montrer que soit $X \oplus C \geqslant_T \emptyset'$, soit $X$ est de degré DNC relativement à $C$.

Indication.– Soit $g$ une fonction DNC. Définir des codes $a_{n,m}$ tels que

$$\Phi_{a_{n,m}}(a_{n,m}) = \Phi_m(C_s, m),$$

pour $s$ le plus petit tel que $n \in \emptyset'[s]$. Montrer que soit il existe $n$ tel que la fonction $m \mapsto g(a_{n,m})$ est DNC relativement à $C$, soit $g \oplus C$ calcule $\emptyset'$. ⋄

Notons que l'exercice précédent implique le critère de complétude d'Arslanov, car si $X$ et $C$ sont de même degré, soit $C \geqslant_T \emptyset'$, soit $C$ est de degré DNC relativement à $C$, ce qui est impossible.

## 4. Fonctions hyperimmunes

Nous abordons maintenant une seconde famille de propriétés calculatoires, basées sur la capacité à calculer des fonctions à croissance rapide. Les fonctions exponentielles, ou même de tours d'exponentielles sont calculables, et n'apportent donc pas de puissance de calcul supplémentaire. Nous parlons ici de fonctions dont la vitesse de croissance rend difficile toute représentation mentale.

Nous avons déjà vu dans la section 4-7 que la capacité à croître plus vite que certaines fonctions permettait de calculer les ensembles $\Delta^0_2$. Nous allons maintenant étudier la puissance de calcul liée aux fonctions qui ne sont dominées par aucune fonction calculable. L'étude de ces fonctions a été initiée par Martin et Miller [157].

**Définition 4.1.** Une fonction $g$ *domine* une fonction $f$ si $g(x) \geqslant f(x)$ pour tout $x \in \mathbb{N}$. Une fonction $f : \mathbb{N} \to \mathbb{N}$ est *hyperimmune* si elle n'est dominée par aucune fonction calculable. ♢

En particulier, les fonctions calculables étant stables par modifications finies, une fonction $f : \mathbb{N} \to \mathbb{N}$ est hyperimmune si pour toute fonction calculable $g : \mathbb{N} \to \mathbb{N}$, $f(x) > g(x)$ pour une infinité de valeurs $x$.

**Exercice 4.2.** Montrer qu'une fonction $f$ est hyperimmune si, et seulement si, il existe pour toute fonction totale calculable $g$ une infinité d'entiers $x$ tels que $f(x) > g(x)$. ◇

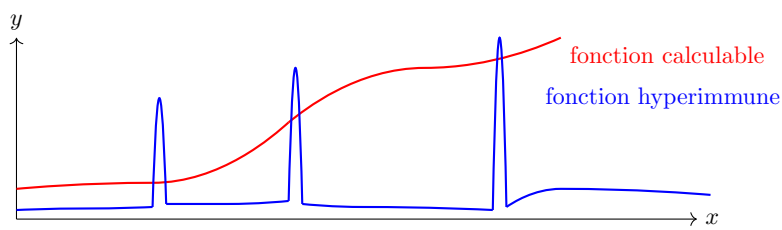



FIGURE 4.3 – *Illustration d'une fonction hyperimmune : elle ne croît pas nécessairement très vite, mais pour toute fonction calculable $f$ elle se situe infiniment souvent au-dessus de $f$.*

Tout comme les degrés DNC, les degrés Turing des fonctions hyperimmunes sont clos par le haut. Un degré Turing est *hyperimmune* s'il contient une fonction hyperimmune, ou de manière équivalente si un de ses éléments calcule une fonction hyperimmune. L'appellation « fonction hyperimmune » provient de la correspondance suivante avec les ensembles hyperimmunes.

**Proposition 4.4.** Un ensemble infini $X$ est hyperimmune si, et seulement si, la fonction $p_X : \mathbb{N} \to \mathbb{N}$ qui à $n$ associe le $n$-ième élément de $X$ est hyperimmune. ⋆

Preuve. $\Rightarrow$. Soit $X$ un ensemble hyperimmune et soit $g : \mathbb{N} \to \mathbb{N}$ une fonction totale calculable. Montrons que $p_X$ n'est pas dominée par $g$. Soit $h$ la fonction donnée par $h(x) = g(x) + x + 1$, et soit $(F_n)_{n\in\mathbb{N}}$ le tableau c. e. que l'on définit par $F_n = [h^{(n)}(0), h^{(n+1)}(0)[$, où $h^{(n)}$ est la $n$-ième itération de $h$, avec $h^{(0)}$ la fonction identité. Par hyperimmunité de $X$, il existe $n$ tel que $F_n \cap X = \emptyset$. Cela signifie que $p_X(h^{(n)}(0)) \geqslant h^{(n+1)}(0) = h(h^{(n)}(0))$. En prenant $x = h^{(n)}(0)$, nous avons $p_X(x) \geqslant h(x) = g(x) + x + 1$, donc $p_X$ n'est pas dominée par $g$.

$\Leftarrow$. Supposons que $p_X$ soit une fonction hyperimmune. Raisonnons par l'absurde, en supposant que l'ensemble $X$ ne soit pas hyperimmune. Autrement dit, il existe un tableau c. e. $(F_n)_{n\in\mathbb{N}}$ tel que $F_n \cap X \neq \emptyset$ pour tout $n$. Alors, la fonction $g : \mathbb{N} \to \mathbb{N}$ définie par $g(n) = \max \bigcup_{i\leqslant n} F_i$ est totale calculable, et domine $p_X$, contredisant l'hyperimmunité de $p_X$. ∎

Il s'ensuit qu'un degré est hyperimmune si, et seulement si, il contient un ensemble hyperimmune.

**Exercice 4.5.** Montrer que les degrés hyperimmunes sont clos par le haut. Autrement dit, si $X$ calcule une fonction hyperimmune, alors le degré Turing de $X$ contient une fonction hyperimmune. ◇

**Exercice 4.6.** Montrer par la méthode des extensions finies qu'il existe un ensemble de degré hyperimmune. ◇

L'existence de degrés non calculables et non hyperimmunes n'est pour le moment pas claire. Nous verrons dans la section suivante que de tels degrés existent bien, même si nous comprendrons plus tard que ce n'est pas donné, dans le sens ou « beaucoup » d'ensembles sont de degrés hyperimmune (voir la proposition 10-3.35 et le théorème 19-3.4). De fait, il va falloir travailler pour en exhiber un qui ne le soit pas. Nous voyons en particulier dès à présent que la construction d'un degré non calculable et non hyperimmune ne peut pas se faire à l'aide de $\emptyset'$. En particulier, des constructions par extensions finies comme vues dans la section 4-8, qui sont toutes effectives en $\emptyset'$, ne pourront pas fonctionner.

**Proposition 4.7 (Martin et Miller [157]).** Tout ensemble $\Delta^0_2$ non calculable est hyperimmune. ⋆

Preuve. Soit $A$ un ensemble $\Delta^0_2$ non calculable, et soit $A_0, A_1, \ldots$ une approximation $\Delta^0_2$ de $A$. Rappelons que la *fonction de calcul* (Définition 4

-7.7) est définie comme la fonction $c_A : \mathbb{N} \to \mathbb{N}$ qui à $x$ associe le plus petit entier $n \geqslant x$ tel que $A_n \upharpoonright_x = A \upharpoonright_x$. En particulier, $c_A \leqslant_T A$. D'après la proposition 4-7.9, toute fonction dominant $c_A$ calcule $A$. Comme $A$ n'est pas calculable, $c_A$ n'est dominée par aucune fonction calculable, autrement dit $c_A$ est hyperimmune. ∎

Nous passons à présent à l'existence de degrés non calculables et non hyperimmunes, que l'on dira *calculatoirement dominés*.

## 5. Degrés calculatoirement dominés

Par clôture des degrés hyperimmunes par le haut, un degré Turing n'est pas hyperimmune si toute fonction $f$ qu'il calcule est dominée par une fonction calculable $g$ (qui dépend de $f$).

**Définition 5.1.** Un ensemble $X$ est *calculatoirement dominé* si pour toute fonction $f \leqslant_T X$ il existe une fonction calculable $g$ dominant $f$. Un degré Turing $\mathbf{d}$ est calculatoirement dominé [a] si tout $X \in \mathbf{d}$ est calculatoirement dominé. ♢

*a.* En anglais, les terminologies « computably dominated » et « hyperimmune-free » coexistent.

Notons que le fait d'être calculatoirement dominé est une propriété de faiblesse et est donc close par le bas dans les degrés Turing. De fait, la propriété inverse — être de degré hyperimmune — est une propriété de force.

L'exemple le plus simple de degré calculatoirement dominé est le degré Turing des ensembles calculables. Le but de cette section est de démontrer l'existence de degrés calculatoirement dominés différents de $\mathbf{0}$. Nous verrons en fait un peu plus tard (théorème 8-5.1) que les degrés calculatoirement dominés sont en quantité indénombrable : il est possible de construire une injection $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ telle que pour tout $X$, l'ensemble $f(X)$ est de degré calculatoirement dominé, et même telle que $f(X)$ et $f(Y)$ soient dans des degrés Turing différents pour $X \neq Y$ (exercice 8-5.3). Le concept au cœur de la construction qui va suivre est celui de *f-arbre*.

**Définition 5.2.** Un *f-arbre* est une fonction totale $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ telle que pour tous $\sigma, \tau \in 2^{<\mathbb{N}}$, $\sigma \preceq \tau$ si, et seulement si, $T(\sigma) \preceq T(\tau)$. ♢

Soit $T$ un $f$-arbre. Notons que l'image de $T$ ($\operatorname{Im} T$) n'est pas close par préfixe en général. Notons également que pour tout $\sigma \in 2^{<\mathbb{N}}$, $T(\sigma 0)$ et $T(\sigma 1)$ sont deux chaînes incompatibles étendant $T(\sigma)$. Seule l'image d'un $f$-arbre $T$ est importante. La structure arborescente du domaine de $T$ induit de manière canonique celle de $\operatorname{Im} T$. Le lecteur peut se reporter à la figure 5.4 pour une représentation graphique d'un f-arbre.

**Définition 5.3.** Soit $T$ un f-arbre. On appelle *nœuds* les éléments de $\operatorname{Im} T$. Un *chemin* de $T$ est une suite $P \in 2^{\mathbb{N}}$ dont une infinité de segments initiaux appartiennent à $\operatorname{Im} T$. On notera $[T]$ l'ensemble des chemins de $T$.♢

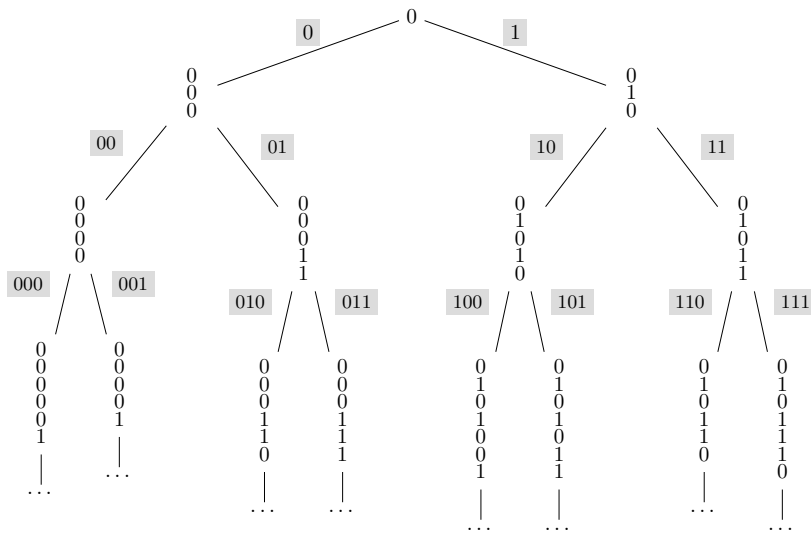



FIGURE 5.4 – *Illustration d'un f-arbre $T$, dont le domaine est représenté par les chaînes en gris clair : $T(\epsilon) = 0$, $T(0) = 000$, $T(1) = 010$, ...*

La figure 5.4, ci-dessus, illustre le fait qu'un f-arbre $T$ induit une injection $f_T : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$, calculable à l'aide du f-arbre : pour $X \in 2^{\mathbb{N}}$, la suite $f_T(X)$ se calcule petit à petit comme étant $T(X\restriction_1) \prec T(X\restriction_2) \prec \dots$ De plus, par les propriétés d'un f-arbre, si $X \neq Y$, alors $f_T(X) \neq f_T(Y)$.

**Définition 5.5.** Un *sous-f-arbre* d'un f-arbre $T$ est un f-arbre $S$ tel que

$$\operatorname{Im} S \subseteq \operatorname{Im} T.$$ ♢

Il s'ensuit que si $S$ est un sous-f-arbre de $T$, alors $[S] \subseteq [T]$. Nous avons à présent les ingrédients nécessaires pour passer à la preuve du théorème annoncé.

**Théorème 5.6 (Martin et Miller [157])**
*Il existe un degré* $\mathbf{d} > \mathbf{0}$ *calculatoirement dominé.*

PREUVE. Nous voulons construire un ensemble $A$ de degré calculatoirement dominé en respectant les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ suivants :

$\mathcal{R}_e : W_e \neq A \qquad \mathcal{S}_e : \Phi_e^A$ total $\Rightarrow \Phi_e^A$ dominé par une fonction calculable.

Nous allons construire une suite infinie de f-arbres calculables $T_0, T_1, T_2, \ldots$ tels que pour tout $e \in \mathbb{N}$,

(1) $T_{e+1}$ est un sous-f-arbre de $T_e$ ;

(2) $|T_e(\epsilon)| \geqslant e$ ;

(3) Pour tout chemin $P \in [T_{2e+1}]$, le contrat $\mathcal{R}_e$ est satisfait ;

(4) Pour tout chemin $P \in [T_{2e+2}]$, le contrat $\mathcal{S}_e$ est satisfait.

**Satisfaction d'un contrat $\mathcal{R}_e$.** Donnons-nous un f-arbre calculable $T$. Comme $\sigma_0{=}T(0)$ et $\sigma_1{=}T(1)$ sont des chaînes incompatibles, il existe $i{\in}\mathbb{N}$ tel que $\sigma_0(i) \neq \sigma_1(i)$. Ainsi, soit $\sigma_0(i) \neq W_e(i)$, soit $\sigma_1(i) \neq W_e(i)$. Supposons que l'on soit dans le premier cas, l'autre étant symétrique. Alors, le sous-f-arbre $S$ de $T$ défini par $S(\rho) = T(0\rho)$ assure que pour tous les chemins $P \in [S]$, $\sigma_0 \preceq P$, donc $P \neq W_e$. De plus, $S$ est calculable en $T$, donc calculable.

**Satisfaction d'un contrat $\mathcal{S}_e$.** Soit $T$ un f-arbre calculable. Nous allons construire un sous-f-arbre calculable $S$ tel que soit $\Phi_e^P$ est partiel pour tout $P \in [S]$, soit $\Phi_e^P$ est total et dominé par une même fonction calculable pour tout $P \in [S]$. Notons qu'en plus de satisfaire le contrat $\mathcal{S}_e$, la fonctionnelle $\Phi_e^P$ sera soit partielle, soit totale quel que soit le chemin $P$. Les deux cas suivants se présentent.

Cas 1. Il existe un nœud $\sigma \in \operatorname{Im} T$ et une entrée $x \in \mathbb{N}$ tels que $\Phi^\tau(x)\uparrow$ pour tout $\tau \in \operatorname{Im} T$ tel que $\tau \succeq \sigma$. Soit $\rho \in 2^{<\mathbb{N}}$ tel que $T(\rho) = \sigma$. Par suite, le sous-f-arbre $S$ de $T$ défini par $S(\mu) = T(\rho\mu)$ assure que pour tous les chemins $P \in [S]$, des segments initiaux $\tau$ de $P$ arbitrairement longs satisferont $\Phi^\tau(x)\uparrow$. Par la propriété de l'usage, il s'ensuit que $\Phi^P(x)\uparrow$.

Cas 2. Pour tout nœud $\sigma \in \operatorname{Im} T$ et toute entrée $x \in \mathbb{N}$, il existe $\tau \in \operatorname{Im} T$ tel que $\sigma \preceq \tau$ et $\Phi^\tau(x)\downarrow$. Nous allons définir $S$, un sous-f-arbre calculable de $T$ tel que pour tout $\rho \in 2^{<\mathbb{N}}$, $\Phi_e^{S(\rho)}(|\rho|)\downarrow$. On calcule $S(\epsilon)$ comme étant le premier nœud de $\operatorname{Im} T$ que l'on trouve tel que $\Phi_e^{S(\epsilon)}(0)\downarrow$. Supposons que l'on a calculé $S(\rho)$. Soit $\mu$ tel que $S(\rho) = T(\mu)$. Nous calculons $S(\rho 0)$ et $S(\rho 1)$ comme suit : pour chaque entier $i < 2$, $S(\rho i)$ est le premier nœud $\tau \in \operatorname{Im} T$ que l'on trouve, avec $\tau \succeq T(\mu i)$ et tel que $\Phi^\tau(|\rho| + 1)\downarrow$. L'algorithme qui recherche les nœuds $S(\rho 0)$ et $S(\rho 1)$ aboutira toujours, par l'hypothèse que nous sommes dans le cas 2. Par construction, $S$ est bien un f-arbre, et $\operatorname{Im} S \subseteq \operatorname{Im} T$. Plus encore, $\Phi_e^P$ est une fonction totale pour tout $P \in [S]$.

Montrons que $\Phi_e^P$ est dominée par une fonction calculable pour tout chemin $P \in [S]$. Soit $g : \mathbb{N} \to \mathbb{N}$ définie par $g(n) = \max\{\Phi_e^{S(\rho)}(n) : |\rho| = n\}$. Alors, pour tous $n \in \mathbb{N}$ et $P \in [S]$, $\Phi_e^P(n) \leqslant g(n)$. Le sous-f-arbre $S$ de $T$ satisfait bien le contrat $\mathcal{S}_e$.

Notons enfin pour satisfaire le point (2) ci-dessus que pour tout f-arbre $T$ calculable et tout $n$, il existe un sous-f-arbre $S$ tel que $S(\epsilon) \geqslant n$. En effet, il suffit de fixer une chaîne $\rho$ de longueur $n$, et de définir $S(\mu) = T(\rho\mu)$. Nous pouvons donc combiner les satisfactions des différents contrats et cette dernière observation pour construire une suite de f-arbres $T_0, T_1, \ldots$ satisfaisant les propriétés (1) (2) (3) et (4) données plus haut.

**Remarque**

Chaque f-arbre $T_e$ de la suite pris indépendamment est calculable, mais la suite $T_0, T_1, \ldots$ n'est pas elle-même calculable.

Afin de terminer la preuve, nous avons besoin du lemme suivant.

**Lemme 5.7.** L'intersection $\bigcap_e [T_e]$ contient exactement un élément. ⋆

Preuve. Comme $\operatorname{Im} T_{e+1} \subseteq \operatorname{Im} T_e$ et comme toute chaîne de $\operatorname{Im} T_e$ est une extension de $T_e(\epsilon)$, il est clair que l'on a $T_0(\epsilon) \preceq T_1(\epsilon) \prec T_2(\epsilon) \prec \ldots$

Par ailleurs, comme $|T_e(\epsilon)| \geqslant e$ , la suite $T_0(\epsilon) \preceq T_1(\epsilon) \prec T_2(\epsilon) \prec \ldots$ converge vers une unique suite infinie $X \in 2^{\mathbb{N}}$. Comme $X$ a une infinité de préfixes dans chaque $T_e$, alors $X \in [T_e]$ pour tout $e$, et donc $X \in \bigcap_e [T_e]$. ■

Soit $A$ l'élément de $\bigcap_e [T_e]$. Pour tout $e \in \mathbb{N}$, comme $A \in [T_{2e+1}]$, alors le contrat $\mathcal{R}_e$ est satisfait pour $A$, donc $W_e \neq A$. De plus, comme $A \in [T_{2e+2}]$, alors le contrat $\mathcal{S}_e$ est satisfait pour $A$, donc si $\Phi_e^A$ est total, $\Phi_e^A$ est dominé par une fonction calculable. Il s'ensuit que $A$ est calculatoirement dominé et non calculable. ■

**Remarque**

Une analyse soigneuse de la construction précédente montre qu'il suffit de l'oracle $\emptyset''$ pour calculer uniformément la suite $(T_e)_{e \in \mathbb{N}}$. En effet, les seules parties non calculables sont les analyses de cas, qui sont des propriétés $\Sigma_2^0$. Ainsi, il existe un degré $\mathbf{d} > \mathbf{0}$ à la fois $\Delta_3^0$ et calculatoirement dominé. Comme nous l'avons vu avec la proposition 4.7, il n'est pas possible d'abaisser cette borne à $\Delta_2^0$.

Nous retrouverons la structure des f-arbres dans le chapitre 14 pour montrer l'existence de degrés Turing minimaux.

**Exercice 5.8. (⋆⋆)** Construire un f-arbre $T$, par la méthode des extensions finies, tel que tout $X, Y \in [T]$ sont dans des degrés Turing différents. ◇

Notons que nous avions annoncé l'existence d'une quantité indénombrable de degrés calculatoirement dominés. Cela sera fait avec le théorème 8-5.1.

Nous donnons à présent quelques équivalences permettant de mieux cerner cette notion.

### 5.1. Réduction truth-table

On suppose ci-après que les fonctionnelles considérées essayent de calculer des ensembles d'entiers et non des fonctions, c'est-à-dire que si $\Phi(Y,n)\downarrow$ pour un certain oracle $Y$ et un certain entier $n$, alors $\Phi(Y,n)\downarrow \in \{0,1\}$ (si $\Phi(Y,n)\downarrow \notin \{0,1\}$, on considérera alors que la fonctionnelle diverge).

Soient une fonctionnelle $\Phi$ et un oracle $X$ tels que $\forall n\ \Phi(X,n)\downarrow \in \{0,1\}$. Étant donné un autre ensemble $Y \neq X$, il n'y a aucune raison pour que l'on ait également $\forall n\ \Phi(Y,n)\downarrow$. Une fonctionnelle totale avec un oracle ne l'est pas forcément avec les autres, et il ne devrait pas être très dur pour le lecteur de construire de tels exemples. Mais de telles fonctionnelles sont-elles nécessaires ? Si l'on suppose $X \geqslant_T Y$, peut-on toujours calculer $Y$ à partir de $X$ via une fonctionnelle totale sur tous les oracles ? Nous allons voir que ce n'est pas forcément le cas via une restriction de la notion de réduction Turing.

**Définition 5.9.** Pour tous ensembles $X, Y \subseteq \mathbb{N}$, on dit que $X$ est *truth-table réductible* à $Y$, et l'on écrit $X \leqslant_{tt} Y$, s'il existe une fonctionnelle $\Phi$ telle que $\Phi(Y) = X$ et telle que $\Phi(Z)$ est totale pour tout oracle $Z$. On écrit $X \equiv_{tt} Y$ si $X \leqslant_{tt} Y$ et $Y \leqslant_{tt} X$. On écrit $X <_{tt} Y$ si $X \leqslant_{tt} Y$ et $Y \not\leqslant_{tt} X$. On appelle *degrés truth-table* les classes d'équivalences de la relation $\equiv_{tt}$. ♢

Remarquons la notation $\Phi(Y) = X$ signifiant $\forall n\ \Phi(Y,n) = X(n)$. Nous verrons plusieurs définitions équivalentes à la réduction truth-table, à commencer par celle justifiant son nom, que l'on peut traduire en français par « réduction par table de vérité ».

**Définition 5.10.** Une réduction par table de vérité est donnée par une suite calculable de paires $(\langle C_{0,n}, C_{1,n}\rangle)_{n\in\mathbb{N}}$ telle que pour tout $n$ l'ensemble $C_{0,n} \cup C_{1,n} \subseteq 2^{<\mathbb{N}}$ contient exactement toutes les chaînes d'une certaine taille $m_n$, et telle que $C_{0,n} \cap C_{1,n} = \emptyset$. L'ensemble $X$ calcule $Y$ via cette réduction si pour tout $n$ on a $Y(n) = i$ ssi $\sigma \in C_{i,n}$ pour un préfixe $\sigma$ de $X$. ♢

Les ensembles $C_{i,n}$ sont les « tables de vérité ». N'importe quel oracle $X$ admet un préfixe dans $C_{0,n} \cup C_{1,n}$. Si le préfixe appartient à $C_{0,n}$, alors $X$ calcule 0 sur l'entrée $n$ ; et, si le préfixe appartient à $C_{1,n}$, alors $X$ calcule 1 sur l'entrée $n$. Voyons à présent les différentes équivalences à la notion de réduction truth-table.

**Théorème 5.11**
*Soient $X, Y$ des ensembles. Les énoncés suivants sont équivalents.*

*(1)* $Y \leqslant_{tt} X$.

*(2)* *$X$ calcule $Y$ via une réduction par table de vérité.*

*(3)* *Il existe une fonctionnelle $\Phi$ et une fonction $b : \mathbb{N} \to \mathbb{N}$ totale calculable telle que $\Phi(X, n)[b(n)]\downarrow = Y(n)$ pour tout $n$.*

Preuve. Montrons (1) $\Rightarrow$ (2). Supposons $\Phi(X) = Y$ via une fonctionnelle $\Phi$ totale sur tous les oracles. Étant donné $n$, on cherche le plus petit temps de calcul $t_n$ tel que, pour un certain $m_n \leqslant t_n$ et pour toute chaîne $\sigma \in 2^{\mathbb{N}}$ de taille $m_n$, on a $\Phi(\sigma, n)[t_n] \downarrow$. Afin de montrer qu'un tel temps de calcul $t_n$ existe forcément pour tout $n$, on doit anticiper un peu sur la définition 8-1.1 *d'arbre* et le lemme 8-1.4 de König à venir. Supposons par l'absurde que, pour un certain $n$, on ne puisse trouver $t_n$. Cela implique en particulier que, pour tout $m$, il existe une chaîne $\sigma$ de taille $m$ telle que $\forall t\ \Phi(\sigma, n)[t]\uparrow$. Par ailleurs, si $\forall t\ \Phi(\sigma, n)[t]\uparrow$ et $\tau \preceq \sigma$, alors aussi $\forall t\ \Phi(\tau, n)[t]\uparrow$. On peut donc construire un arbre infini $T$ tel que $\sigma \in T$ implique $\forall t\ \Phi(\sigma, n)[t]\uparrow$. D'après le lemme de König, $T$ contient un chemin infini $Y$, qui est donc tel que $\Phi(Y, n)\uparrow$, ce qui contredit le fait que $\Phi$ soit totale sur tous ses oracles. On peut donc à chaque étape trouver $t_n$ et $m_n \leqslant t_n$, avec l'ensemble $C_{0,n}$ des chaînes de tailles $m_n$ sur lesquelles le calcul renvoie 0 en $t_n$ étapes, et l'ensemble $C_{1,n}$ des chaînes de tailles $m_n$ sur lesquelles le calcul renvoie 1 en $t_n$ étapes.

Montrons (2) $\Rightarrow$ (3). Étant donné une réduction « table de vérité » donnée par la suite calculable $(\langle C_{0,n}, C_{1,n}\rangle)_{n\in\mathbb{N}}$, pour toute entrée $n$ on peut borner le temps de calcul que la fonctionnelle met pour s'arrêter sur $n$ avec n'importe quel oracle : il s'agit simplement du temps nécessaire pour produire le calcul de $\langle C_{0,n}, C_{1,n}\rangle$.

Montrons (3) $\Rightarrow$ (1). Soit $\Phi$ une fonctionnelle, et soit $b : \mathbb{N} \to \mathbb{N}$ une fonction totale calculable telle que

$$\Phi(Y, n)[b(n)]\downarrow = X(n), \qquad \text{pour tout } n.$$

Alors, on construit la fonctionnelle $\Psi$ qui sur tous les oracles $Z$ et sur toute entrée $n$ lance le calcul de $\Phi(Z, n)$ en $b(n)$ étapes. Si le calcul renvoie une valeur en $b(n)$ étapes, alors $\Psi$ renvoie cette valeur, sinon $\Psi$ renvoie 0. Le résultat du calcul est le même entre $\Phi$ et $\Psi$ sur l'oracle $Y$, mais $\Psi$ est maintenant totale sur tous les oracles. ∎

Nous montrons à présent que les ensembles $X$ pour lesquels $X \geqslant_T Y$ implique $X \geqslant_{tt} Y$ sont exactement les ensembles calculatoirement dominés.

**Théorème 5.12 (Jockusch [97], Martin (non publié))**
*Un ensemble $X$ est calculatoirement dominé ssi $Y \leqslant_T X \Leftrightarrow Y \leqslant_{tt} X$ pour tout $Y \in 2^{\mathbb{N}}$.*

Preuve. Supposons $X$ calculatoirement dominé. Supposons $X \geqslant_T Y$ via la fonctionnelle $\Phi$. Soit $f : \mathbb{N} \to \mathbb{N}$ telle que $\Phi(X, n)[f(n)] \downarrow = Y(n)$ pour tout $n$. Notons que $f$ est une fonction $X$-calculable. Il y a donc une fonction calculable $g > f$. On a ainsi $\Phi(X, n)[g(n)] \downarrow = Y(n)$. D'après le théorème 5.11, on a donc $X \geqslant_{tt} Y$.

Supposons maintenant que pour tout $Y$ on ait $Y \leqslant_T X \Leftrightarrow Y \leqslant_{tt} X$. Alors, également pour toute fonction $f : \mathbb{N} \to \mathbb{N}$, on a $f \leqslant_T X \Leftrightarrow f \leqslant_{tt} X$, via la représentation canonique de $f$ par une suite de $2^{\mathbb{N}}$. Soit $f \leqslant_T X$ ; montrons que $f$ est dominée par une fonction calculable $g$. Par hypothèse, $f \leqslant_{tt} X$, donc par le théorème 5.11, il existe une fonctionnelle $\Phi$ et une fonction totale calculable $b : \mathbb{N} \to \mathbb{N}$ telle que $\Phi(X, n)[b(n)]\downarrow = f(n)$ pour tout $n$. On peut supposer sans perte de généralité que si $\Phi(X, n)[b(n)]\downarrow$, alors l'usage du calcul est inférieur à $b(n)$ (si ce n'est pas le cas, on peut ralentir le calcul pour que cela le soit), donc $\Phi(X\restriction_{b(n)}, n)[b(n)]\downarrow$. Soit $g : \mathbb{N} \to \mathbb{N}$ la fonction qui pour une entrée $n$, exécute $\Phi(\sigma, n)[b(n)]$ pour tout $\sigma \in 2^{<\mathbb{N}}$ de longueur $b(n)$, et renvoie le maximum des valeurs obtenues. En particulier,

$$g(n) \geqslant \Phi(X\restriction_{b(n)}, n)[b(n)] = f(n).$$

La fonction $g$ domine $f$. Il s'ensuit que $X$ est de degré calculatoirement dominé. ∎

Nous avons à présent trois notions de réduction : la réduction many-one, la réduction truth-table et la réduction Turing. La proposition suivante récapitule certains résultats vus jusqu'ici, qui attestent qu'aucune ne coïncide avec une autre.

**Proposition 5.13.** Pour tous $X, Y$, on a

$$X \leqslant_m Y \Rightarrow X \leqslant_{tt} Y \Rightarrow X \leqslant_T Y.$$

Aucune implication inverse n'est vraie dans le cas général. ⋆

Preuve. Les implications sont claires. Montrons qu'aucune implication réciproque ne tient. L'ensemble $\mathbb{N} \setminus \emptyset'$ est $tt$-réductible à $\emptyset'$ mais non $m$-réductible à $\emptyset'$, car il serait alors $\Sigma^0_1$ d'après la proposition 5-4.3, et donc $\emptyset'$ serait calculable ce qui est faux.
L'existence d'ensembles $X, Y$ tels que $X \geqslant_T Y$ mais $X \not\geqslant_{tt} Y$ est une conséquence du théorème 5.12 et du fait que des degrés non calculatoirement dominés existent (par exemple $\mathbf{0}'$). ∎

# 6. Théorème de domination de Martin

Les fonctions hyperimmunes sont par définition les fonctions qui ne sont dominées par aucune fonction calculable. Il est naturel de se poser la question de la puissance de calcul des fonctions qui dominent toutes les fonctions calculables. Bien entendu, aucune fonction $f : \mathbb{N} \to \mathbb{N}$ ne domine toutes les fonctions calculables, car la fonction constante $f(0) + 1$ n'est pas dominée par $f$. On peut cependant affaiblir la propriété, et se poser la question de la puissance calculatoire d'une fonction $f : \mathbb{N} \to \mathbb{N}$ telle que pour toute fonction calculable $g : \mathbb{N} \to \mathbb{N}$, $f$ domine $g$ *presque partout*, c'est-à-dire partout sauf pour un nombre fini de valeurs.

**Notation**

On utilisera les notations $\forall^\infty m$ et $\exists^\infty m$ pour signifier respectivement $\exists n\ \forall m > n$ et $\forall n\ \exists m > n$. Ainsi, $\forall^\infty m$ signifie « pour presque tout $m$ », et $\exists^\infty m$ signifie « pour une infinité de $m$ ».

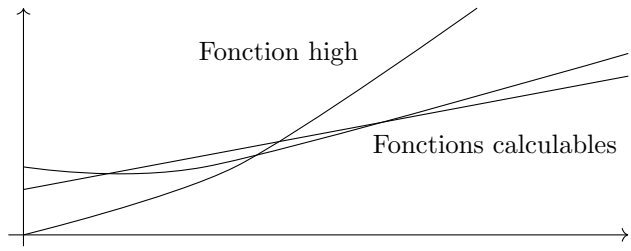



FIGURE 6.1 – *Illustration d'une fonction high, qui pour toute fonction calculable $f$, est toujours au-dessus de $f$ à partir d'une certaine valeur*

Le théorème de domination de Martin ((1) ⇔ (2) dans le théorème suivant [148]) donne une magnifique caractérisation des degrés Turing de ces fonctions. L'équivalence (2) ⇔ (3) montrée par Jockusch [98] est venue plus tard et présente elle aussi son intérêt. Rappelons qu'un ensemble $A$ est high si $A' \geqslant_T \emptyset''$ (voir la définition 4-10.1).

**Théorème 6.2**
*Soit $A \subseteq \mathbb{N}$ un ensemble. Les énoncés suivants sont équivalents.*

*(1) $A$ est high.*

*(2) $A$ calcule une fonction $g : \mathbb{N} \to \mathbb{N}$ qui domine presque partout toutes les fonctions calculables. Cela signifie que pour toute fonction calculable $f$, on a $\forall^\infty n\ f(n) \leqslant g(n)$.*

*(3) $A$ calcule une liste $(X_n)_{n\in\mathbb{N}}$ contenant (éventuellement avec répétitions) exactement les ensembles calculables.*

Preuve. Montrons (1) ⇒ (2). Supposons que $A$ soit high. On a donc une description $\Delta^0_2(A)$ de $\emptyset''$, c'est-à-dire une fonction $A$-calculable $f : \mathbb{N}^2 \to \mathbb{N}$ telle que $\lim_{s\to\infty} f(n,s) = \emptyset''(n)$ pour tout $n$. Notons qu'être le code d'une fonction totale est une propriété $\Pi^0_2$. On peut en particulier, en utilisant le fait que $\emptyset''$ soit $\Sigma^0_2$-complet (voir la proposition 5-5.3), calculer pour tout $e$ un code $a_e$ tel que $\emptyset''(a_e) = 0$ ssi $\Phi_e$ est une fonction totale.

On définit $g$ de la manière suivante : sur l'entrée $t$, pour toute fonctionnelle $\Phi_e$ pour $e \leqslant t$, on cherche le plus petit temps de calcul $s \geqslant t$ tel que $f(a_e, s) = 1$ ou tel que $\Phi_e(t)[s]\downarrow$. Notons qu'un des deux événements arrive forcément : soit $\Phi_e$ est totale et donc $\Phi_e(t)\downarrow$, soit $\Phi_e$ est partielle et donc $\lim_{s\to\infty} f(a_e, s) = \emptyset''(a_e) = 1$. Dans le premier cas, on définit $v_{t,e} = 0$, et dans le second, $v_{t,e} = \Phi_e(t)$. On définit finalement $g(t) = \sum_{e\leqslant t} v_{t,e}$.

Il est clair que pour toute fonction totale $\Phi_e$, à partir du plus petit $t \geqslant e$ tel que $f(a_e, s) = 0$ pour $s \geqslant t$, on aura $g(s) \geqslant \Phi_e(s)$ pour tout $s \geqslant t$. Donc, $g$ domine presque partout toutes les fonctions calculables.

Montrons (2) ⇒ (3). Supposons à présent que $A$ calcule une fonction $g$ qui domine presque partout toute fonction calculable. On utilise le fait que si $\Phi_e$ est totale à valeur dans $\{0,1\}$, alors la fonction calculable $t : \mathbb{N} \to \mathbb{N}$ qui sur $n$ renvoie le plus petit temps de calcul tel que $\Phi_e(n)[t(n)]\downarrow$, pour tout $n$, est dominée par $g$ presque partout.

Pour chaque fonctionnelle $\Phi_e$, on calcule l'ensemble $Y_e$ comme étant

$$Y_e(n) = \Phi_e(n)[g(n)] \text{ si } \Phi_e(n)[g(n)]\downarrow \in \{0,1\}, \quad \text{et } Y_e(n) = 0 \text{ sinon.}$$

La liste $(Y_e)_{e\in\mathbb{N}}$ contient donc, à modification finie près, exactement les ensembles calculables. Pour les obtenir tous, on calcule en fin de compte la suite $(X_n)_{n\in\mathbb{N}}$ comme étant toutes les modifications finies possibles des ensembles $Y_e$.

Montrons (3) ⇒ (1). Le lecteur peut s'aider de la figure 6.4 pour la compréhension de cette implication. Supposons que $A$ calcule une liste $(X_n)_{n\in\mathbb{N}}$ contenant (éventuellement avec répétitions) exactement les ensembles calculables. Soit $P = \{e \ : \ \forall x_1\, \exists x_2\ R(e, x_1, x_2)\}$ un ensemble $\Pi^0_2$ quelconque. Montrons que $P$ est $\Sigma^0_2(A)$. Pour cela, on définit uniformément pour tout $e$ une fonction partielle calculable $f_e : \mathbb{N} \to \{0,1\}$ telle que :

(a) $e \in P$ implique que $f_e$ est une fonction totale calculable ;

(b) $e \notin P$ implique que $f_e$ est une fonction partielle qui ne peut pas être complétée en une fonction totale calculable.

Soit $e$ fixé. On décrit un processus uniforme en $e$. À l'étape de calcul $t$, pour toute valeur $n \leqslant t$ et telle que $f_e$ ne n'arrête pas pour le moment sur $n$, on procède comme suit : si $\Phi_n(n)[t]\downarrow \neq 0$, on définit $f_e(n) = 0$.

Sinon, si $\Phi_n(n)[t]\downarrow = 0$, on définit $f_e(n) = 1$. Sinon, si pour tout $k \leqslant n$ il existe $m_k \leqslant t$ tel que $R(e, k, m_k)$, alors on définit $f_e(n) = 0$.

Le processus est clairement calculable. Montrons (a). Supposons $e \in P$. Alors, pour tout $n$, il existe un plus petit $t$ tel que pour tout $k \leqslant n$ il existe $m_k \leqslant t$ pour lequel $R(e, k, m_k)$. Quand cela arrive, alors $f_e(n)$ prend une valeur à l'étape $t$ si elle n'en a pas prise jusqu'ici. Donc, $f_e$ est totale. Montrons à présent l'assertion (b). Supposons $e \notin P$. Soit $n$ le plus grand entier tel que pour tout $k \leqslant n$ il existe $m_k$ pour lequel $R(e, k, m_k)$. Pour $m > n$, $f_e(m)$ s'arrête ssi $\Phi_m(m)$ s'arrête, auquel cas $f_e(m) \neq \Phi_m(m)$. Supposons par l'absurde que $f_e$ a une complétion calculable. Alors, par le lemme de remplissage (le lemme 3-5.1), elle a une complétion calculable de code $a > n$. Dans ce cas, $f_e(a) = \Phi_a(a)$, ce qui contredit la définition de $f_e$. On a donc (b).

Il s'ensuit que $P$ peut s'écrire comme un ensemble $\Sigma^0_2(A)$ de la manière suivante.

$$P = \{e \ : \ \exists n \ \forall m \ \forall t \ f_e(m)[t]\uparrow \vee f_e(m)[t]\downarrow = X_n(m)\}.$$

En effet, si $e \in P$, alors $f_e$ est calculable, et coïncide ainsi avec un certain $X_n$. À l'inverse, si $e \notin P$, alors $f_e$ n'a pas de complétion calculable, et donc pas de complétion dans $(X_n)_{n\in\mathbb{N}}$. Comme $P$ est $\Sigma^0_2(A)$ et $\overline{P}$ est par définition $\Sigma^0_2$, on a donc que $P$ est $\Delta^0_2(A)$, et $P$ est dès lors $A'$-calculable. Il suffit d'appliquer cela pour $P = \mathbb{N} \setminus \emptyset''$ pour obtenir que $\emptyset''$ est $\Delta^0_2(A)$, et donc $A'$-calculable. ∎

L'implication (3) ⇒ (1) du théorème 6.2 est loin d'être évidente. Nous espérons que le lecteur saura apprécier l'argument, dont la subtile complexité est caractéristique du travail de Jockusch. L'exercice suivant donne des caractérisations alternatives similaires, plus simples à démontrer.

**Exercice 6.3. (⋆)** Montrer les équivalences suivantes :

(1) $A$ calcule une fonction qui domine presque partout toute fonction calculable ;

(2) $A$ calcule une suite $(f_n)_{n\in\mathbb{N}}$ contenant toutes les fonctions calculables de $\mathbb{N}$ vers $\mathbb{N}$ ;

(3) $A$ calcule une suite $(X_n)_{n\in\mathbb{N}}$ ne contenant que des ensembles infinis, et contenant tous les ensembles calculables infinis. ◇

Discutons un peu de la caractérisation (1) ⇔ (2) du théorème 6.2, illustrée par la figure 6.1. Le fait de pouvoir calculer une fonction de $\mathbb{N}$ dans $\mathbb{N}$ à très forte croissance est susceptible de donner beaucoup de puissance de calcul. Ainsi, par exemple, comme nous l'avons vu, calculer une fonction qui borne le temps d'arrêt des programmes informatiques permet de calculer $\emptyset'$.

| $f_e$ | $X_0$ | $X_1$ | $X_2$ | $\cdots$ |
|---|---|---|---|---|
| $f_e(0)$ | $X_0(0)$ | $X_1(0)$ | $X_2(0)$ | |
| $f_e(1)$ | $X_0(1)$ | $X_1(1)$ | $X_2(1)$ | |
| $\uparrow$ | $X_0(2)$ | $X_1(2)$ | $X_2(2)$ | |
| $\uparrow$ | $X_0(3)$ | $X_1(3)$ | $X_2(3)$ | |
| $f_e(4)$ | $X_0(4)$ | $X_1(4)$ | $X_2(4)$ | |
| $\uparrow$ | $X_0(5)$ | $X_1(5)$ | $X_2(5)$ | |
| $f_e(6)$ | $X_0(6)$ | $X_1(6)$ | $X_2(6)$ | |
| $\uparrow$ | $X_0(7)$ | $X_1(7)$ | $X_2(7)$ | |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | |

FIGURE 6.4 – *Illustration de la preuve (3) $\Rightarrow$ (1) du théorème 6.2 : si $e \notin P$, on construit une fonction partielle calculable $f_e : \mathbb{N} \to \{0,1\}$ qui n'a pas de complétion totale calculable. Dans l'illustration, la fonction $f_e$ ne peut être égale à aucun des ensembles $X_0, X_1, \ldots$ sur toutes les valeurs pour lesquelles elle est définie. En effet, la liste $(X_n)_{n\in\mathbb{N}}$ ne contient que des ensembles calculables. Dans le cas inverse, la fonction $f_e$ est totale calculable, et est donc égale à au moins un des ensembles $X_i$, puisque la liste $(X_n)_{n\in\mathbb{N}}$ contient tous les ensembles calculables.*

Le théorème précédent indique qu'il y a un premier niveau entre les fonctions permettant de calculer l'arrêt simplement parce qu'elles croissent très vite, et les fonctions de croissance calculable : il existe d'après la proposition 4-10.2 des ensembles high ne calculant pas l'arrêt, et donc des fonctions de croissance « intermédiaire ». Nous verrons également que plus une fonction croît rapidement, plus elle a une puissance de calcul importante. Une fonction qui croît suffisamment vite pourra calculer $\emptyset''$, une qui croît encore plus vite pourra calculer $\emptyset'''$, etc. Nous verrons malgré tout avec le théorème 29-5.4 qu'il existe une limite à la puissance de calcul que confère une croissance rapide. La classe des ensembles calculables par n'importe quelle fonction qui croît « suffisamment » vite a une caractérisation précise et reste malgré tout dénombrable.

**Exercice 6.5. (⋆⋆)** La notion d'ensemble c. e. maximal fut introduite dans l'exercice 3-7.13. Un ensemble c. e. $X$ est maximal si $\mathbb{N} \setminus X$ est infini, et si tout ensemble c. e. $Y \supseteq X$ est tel que $Y \setminus X$ est fini ou tel que $\mathbb{N} \setminus Y$ est fini. Soit $X$ un ensemble c. e. maximal ; montrer que $X$ est high. ◇

# 7. Degrés High ou DNC

Nous terminons ce chapitre par un résultat qui combine degré high et DNC, afin d'obtenir une caractérisation naturelle en termes de puissance de calcul : la possibilité de calculer une fonction qui diffère presque partout de toute fonction calculable.

**Théorème 7.1 (Kjos-Hanssen, Merkle et Stephan [113])**
*Soit $X \in 2^{\mathbb{N}}$. Les énoncés suivants sont équivalents :*

*(1) $X$ est de degré high ou DNC ;*

*(2) $X$ calcule une fonction qui est différente presque partout de toute fonction calculable.*

Preuve. Montrons (1) $\Rightarrow$ (2). Supposons pour commencer que $X$ est high. Soit $g \leqslant_T X$ une fonction qui domine presque partout toute fonction calculable. Alors, également, $m \mapsto g(m) + 1$ est différente presque partout de toute fonction calculable. Supposons à présent que $X$ est DNC. Pour chaque $n$, on calcule le code $e_n$ tel que $W_{e_n}$ énumère toutes les valeurs $\Phi_e(n)$ pour $e \leqslant n$ pour lesquelles $\Phi_e(n)\downarrow$. D'après le théorème 2.6, $X$ calcule une fonction $h : \mathbb{N}^2 \to \mathbb{N}$ telle que pour tout $e, n \in \mathbb{N}$, si $|W_e| \leqslant n$, alors $h(e,n) \notin W_e$. Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction $X$-calculable définie par $f(n) = h(e_n, n)$. On a alors $f(n) \notin W_{e_n}$ pour tout $n$. Ainsi, $f(n)$ est différent de $\Phi_0(n), \Phi_1(n), \ldots, \Phi_n(n)$ pour tout $n$, donc $f$ diffère presque partout de toute fonction calculable (et même de toute fonction partielle calculable s'arrêtant sur une infinité de valeurs).

Montrons (2) $\Rightarrow$ (1). Si $X$ est high, il n'y a rien à vérifier.
Supposons donc que $X$ n'est pas high. Soit $g \leqslant_T X$ telle que $\Phi_e$ totale implique $\forall^\infty m\ g(m) \neq \Phi_e(m)$. Nous prétendons avoir également

$$\forall^\infty e\ g(e) \neq \Phi_e(e).$$

Supposons par l'absurde le contraire. Soit $f$ la fonction $X$-calculable qui sur $n$ renvoie le plus petit temps de calcul $t$ tel que $g(m) = \Phi_m(m)[t]\downarrow$ pour un entier $m > n$. Comme $X$ est non high, il existe une fonction calculable $b$ telle que $\exists^\infty n\ b(n) \geqslant f(n)$. Notons que l'on peut supposer sans perte de généralité que $b(n) \leqslant b(n+1)$.

On définit à présent la fonction totale calculable $h$ par $h(n) = \Phi_n(n)[b(n)]$ si $\Phi_n(n)[b(n)]\downarrow$, et $h(n) = 0$ sinon. Supposons à présent $b(n) > f(n)$. Par définition de $f$, il existe une valeur $m > n$ telle que $\Phi_m(m)[f(n)]\downarrow = g(m)$, et donc telle que $\Phi_m(m)[b(n)]\downarrow = g(m)$. Comme $b(m) \geqslant b(n)$, on aura

$$h(m) = \Phi_m(m)[b(m)]\downarrow = g(m).$$

Comme on peut recommencer l'argument pour des valeurs de $n$ arbitrairement grandes, on aura $h(m) = g(m)$ pour une infinité de $m$. Cela contredit

le fait que $g$ diffère presque partout de toute fonction calculable. Donc, nous avons $\forall^\infty e\ g(e) \neq \Phi_e(e)$. Il suffit alors de modifier un nombre fini de valeurs de $g$ pour obtenir une fonction DNC. ∎

Notons pour donner tout son éclat au théorème précédent qu'il est possible de construire des degrés DNC qui ne sont pas high (en combinant le corollaire 18-4.3 avec le corollaire 19-3.9 ou plus simplement en considérant le corollaire 8-6.6) tout comme des degrés high qui ne sont pas DNC (voir le corollaire 10-3.34).

Chapitre 8

# Classes $\Pi^0_1$ et degrés PA

Nous nous sommes jusqu'à présent concentrés principalement sur l'étude des ensembles d'entiers naturels pris individuellement, ou de manière équivalente, des fonctions des entiers vers les entiers. Nous allons maintenant nous tourner vers l'étude de *classes* d'ensembles ou de fonctions, c'est-à-dire d'ensembles d'ensembles d'entiers.
Nous avons déjà vu de nombreuses classes d'ensembles, notamment la classe des ensembles de degré high $\{X \in 2^{\mathbb{N}} : \ X' \geqslant_T \emptyset''\}$, ou celle des ensembles de degré low $\{X \in 2^{\mathbb{N}} : \ X' \equiv_T \emptyset'\}$.

**Ensemble vs classe**

Afin de distinguer les ensembles d'entiers des sous-ensembles de l'espace de Cantor, nous les appellerons respectivement « ensembles » et « classes » quand nous sommes dans un contexte portant sur l'étude des ensembles $X \in 2^{\mathbb{N}}$ ou des classes $\mathcal{A} \subseteq 2^{\mathbb{N}}$.

Les classes que nous considérerons seront définies par des prédicats, et l'étude de la complexité de ces prédicats permettra de déduire des informations sur les éléments que la classe contient.

L'étude des classes va rapidement se montrer centrale dans l'étude des ensembles d'entiers. On commence ici par les classes de la complexité la plus simple possible : les ouverts et fermés effectifs du Cantor. L'étude de classes de complexité supérieure sera poursuivie dans le chapitre 17. Malgré la simplicité apparente des ouverts et fermés, nous verrons rapidement le large éventail de possibilités et la grande richesse qu'ils renferment.

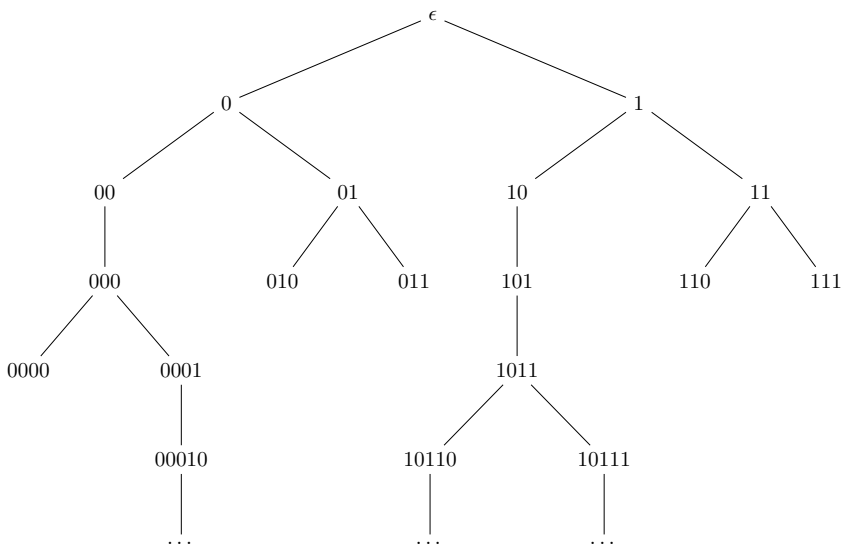



FIGURE 1.2 – *Illustration d'un arbre. La racine $\epsilon$ est la chaîne vide. Chaque nœud admet éventuellement un successeur gauche, un successeur droit, les deux — auquel cas il est branchant — ou aucun des deux — auquel cas c'est une feuille.*

# 1. Arbres binaires

Nous avons défini la notion de f-arbre afin de montrer l'existence d'un degré calculatoirement dominé non calculable (le théorème 7-5.6). Nous introduisons ici une notion similaire et d'une certaine manière plus primitive, à savoir tout simplement les arbres.

**Définition 1.1.** Un ensemble $T \subseteq 2^{<\mathbb{N}}$ est un *arbre* si $T$ est clos par préfixe, c'est-à-dire pour tous $\sigma \in T$ et $\tau \preceq \sigma$, alors $\tau \in T$. ♢

Étant donné un arbre $T \subseteq 2^{<\mathbb{N}}$, on appelle un élément $\sigma \in T$ un *nœud* de l'arbre. Un nœud est *branchant* si $\sigma 0, \sigma 1 \in T$. Dans le cas inverse, il est *non branchant*. On considérera par une sorte de convention sur la représentation mentale que l'on peut avoir d'un arbre, qu'une extension $\sigma 0$ de $\sigma$ « va à gauche dans l'arbre » alors qu'une extension $\sigma 1$ de $\sigma$ « va à droite ». Si $\sigma 0 \in T$, alors $\sigma 0$ est un *successeur gauche* de $\sigma$. Si $\sigma 1 \in T$, alors $\sigma 1$ est un *successeur droit de* $\sigma$. Un nœud sans successeur sera appelé une *feuille*.

**Définition 1.3.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre. Un *chemin* à travers l'arbre $T$ est une suite $P \in 2^{\mathbb{N}}$ telle que $P\restriction_n \in T$ pour tout $n \in \mathbb{N}$. On dénote par $[T]$ la classe des chemins de $T$. ♢

Intuitivement, un chemin $P$ peut être vu comme une suite d'instructions binaires, nous indiquant littéralement « un chemin » à suivre à travers l'arbre. Un bit à 0 dans le chemin nous indique de continuer notre parcours en suivant le successeur gauche et un bit à 1 nous indique de suivre le successeur droit. Nous ne considérons que des chemins infinis.

**Remarque**

Un chemin n'est pas représenté comme un ensemble de nœuds de l'arbre. Il existe cependant une bijection calculable entre un chemin $P$ et l'ensemble $\{P\restriction_n : n \in \mathbb{N}\}$. Représenter un chemin comme une suite binaire infinie est donc principalement un choix conventionnel qui s'avérera très utile par la suite.

Si $T$ est un arbre fini, c'est-à-dire qui n'a qu'un nombre fini de nœuds, alors $[T]$ est forcément l'ensemble vide. Qu'en est-il de la réciproque ? Il s'agit d'un outil central sur les arbres : le lemme de König, qui stipule que tout arbre infini à branchement fini admet un chemin infini. Notez que les arbres $T \subseteq 2^{<\mathbb{N}}$ sont nécessairement 2-branchants. On a affaire dans ce cas au lemme de König *faible*.

**Lemme 1.4 (Lemme de König faible).** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre tel que $|T| = \infty$. Alors, $[T]$ est non vide. ⋆

Preuve. On construit un chemin $X$ par récurrence sur $n$. Comme $T$ est infini, par le principe des tiroirs il existe $i \in \{0,1\}$ et une infinité de nœuds $\sigma \in T$ qui étendent $i$ (c'est-à-dire avec $i \prec \sigma$). On définit $X(0) = i$. Supposons que $\tau = X(0)X(1)\dots X(n)$ soit défini avec $\tau \in T$ et tel qu'il y a une infinité de nœuds $\sigma \in T$ pour lesquels $\tau \preceq \sigma$. Par le principe des tiroirs, il existe $i \in \{0,1\}$ et une infinité de nœuds $\sigma \in T$ qui étendent $\tau i$. On définit $X(n+1) = i$.

Par récurrence sur $n$, on définit donc de cette manière un ensemble $X$ tel que $X\restriction_n \in T$ pour tout $n$. ∎

Le lemme de König peut paraître trivial au premier abord. Le lecteur, à l'intuition bien affûtée et à l'aise avec la manipulation d'objets infinis, se sera éventuellement demandé s'il y avait vraiment besoin de ranger cet énoncé dans un lemme. Nous allons voir que malgré les apparences, ce lemme n'est pas aussi trivial que cela. Bien que simple, il constitue un outil central, particulièrement intéressant de par son contenu calculatoire.

### 1.1. Arbres calculables

Un arbre $T \subseteq 2^{<\mathbb{N}}$ est calculable si l'ensemble $T$ est calculable, autrement dit s'il existe une procédure pour décider si un nœud appartient à l'arbre ou non. Au cours de ce chapitre, nous allons tenter de répondre à la question suivante.

**Question 1.5.** Étant donné un arbre calculable infini, quelle est la puissance de calcul nécessaire pour calculer un chemin infini de $T$ ? ⋆

La preuve du lemme de König fournit une construction claire : quand on a calculé un préfixe $\tau$ de notre chemin infini, on détermine le prochain bit comme étant 0 si l'arbre contient une infinité de nœuds qui étendent $\tau 0$ et comme étant 1 sinon. Le problème est qu'il n'est *a priori* pas possible de savoir de manière calculable si une infinité de nœuds étendent $\tau 0$ : c'est une question pour laquelle l'arrêt des programmes informatiques semble nécessaire. Cela nous conduit à définir la notion de nœud extensible.

**Définition 1.6.** Un nœud $\sigma$ d'un arbre $T \subseteq 2^{<\mathbb{N}}$ est *extensible* dans $T$ si l'ensemble $\{\tau \in T : \tau \succeq \sigma\}$ est infini. ♢

Autrement dit, un nœud $\sigma$ est extensible dans un arbre si le sous-arbre des nœuds compatibles avec $\sigma$ est infini. Souvenons-nous de la notation $[\sigma]$ qui dénote la classe des ensembles $X$ ayant $\sigma$ comme préfixe. Par le lemme de König, un nœud $\sigma$ est extensible dans $T$ si et seulement si $[\sigma] \cap [T] \neq \emptyset$. Notons que dans tout arbre infini, la racine $\epsilon$ est un nœud extensible, et que si $\sigma$ est extensible, alors au moins un nœud parmi $\sigma 0$ et $\sigma 1$ l'est également.

L'exercice suivant montre que les nœuds extensibles sont suffisants pour décrire l'ensemble des chemins d'un arbre.

**Exercice 1.7.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre, et soit $S$ l'ensemble des nœuds extensibles dans $T$. Montrer que $S$ est un arbre, et que $[T] = [S]$. ◇

Il s'ensuit de la définition de nœud extensible que les feuilles ne sont pas extensibles. En revanche, si l'ensemble des feuilles d'un arbre calculable est décidable, ce n'est pas en général le cas de l'ensemble des nœuds extensibles.

**Exercice 1.8.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre calculable infini ne contenant que des nœuds extensibles. Montrer alors que $T$ contient un chemin infini calculable. ◇

De manière générale, déterminer si un ensemble calculable est infini ou non demande l'oracle $\emptyset''$. Dans le cas des arbres, nous pouvons exploiter la clôture par préfixe, pour ramener la complexité de l'oracle à $\emptyset'$. La notation $2^n$ de la proposition suivante désigne l'ensemble des chaînes de taille $n$.

**Proposition 1.9.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre calculable. L'ensemble de ses nœuds extensibles est $\Pi^0_1$. ⋆

Preuve. Les arbres étant clos par le bas, $\{\tau \in T : \tau \succeq \sigma\}$ est infini ssi $\forall n > |\sigma|\ \exists \tau \in 2^n$ tel que $\tau \succeq \sigma$ et $\tau \in T$, ce qui est un prédicat $\Pi^0_1$. Ainsi, l'ensemble des nœuds extensibles de $T$ est l'ensemble $\Pi^0_1$ suivant.

$$\{\sigma \in T : \forall n > |\sigma|\ \exists \tau \in 2^n \text{ tel que } \tau \succeq \sigma \text{ et } \tau \in T\}$$ ∎

Par passage au complémentaire, l'ensemble des nœuds non extensibles d'un arbre calculable est $\Sigma^0_1$, ce qui fait que si un nœud est non extensible, on finira par s'en rendre compte au bout d'un temps fini. Nous allons voir comment utiliser la notion de nœud extensible pour créer des arbres calculables infinis n'ayant pas de chemin calculable. Dans la construction d'un arbre calculable, on doit pouvoir décider en un temps fini si un nœud y appartient ou non. Il est en revanche possible de reporter à plus tard la décision de rendre ou non un nœud extensible, en lui ajoutant par défaut des descendants au cours du temps, jusqu'à décider à un instant $t$ de cesser de lui en rajouter pour le rendre non extensible. Cette technique que l'on appelle « l'astuce du temps » permet de montrer le résultat suivant.

**Proposition 1.10.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre $\Pi^0_1$. Il existe un arbre calculable $S \subseteq 2^{<\mathbb{N}}$ tel que $[T] = [S]$. ⋆

Preuve. Soit $(T_n)_{n\in\mathbb{N}}$ une approximation $\Pi^0_1$ de $T$, c'est-à-dire une suite uniformément calculable d'ensembles décroissante par la relation d'inclusion (avec $T_{n+1} \subseteq T_n$) telle que $\bigcap_n T_n = T$.

Soit $S = \{\sigma \in 2^{<\mathbb{N}} : \forall \tau \preceq \sigma\ \tau \in T_{|\sigma|}\}$. L'ensemble $S$ est calculable. Montrons que $S$ est clos par préfixe. Soit $\sigma \in S$, et soit $\rho \preceq \sigma$. Par définition de $S$, $\forall \tau \preceq \sigma\ \tau \in T_{|\sigma|}$. Comme $|\rho| \leqslant |\sigma|$, $T_{|\rho|} \supseteq T_{|\sigma|}$, donc $\forall \tau \preceq \sigma\ \tau \in T_{|\rho|}$. En particulier, pour tout $\tau \preceq \rho$, également $\tau \preceq \sigma$, donc $\tau \in T_{|\rho|}$. Ainsi, par définition de $S$, $\rho \in S$.

Montrons maintenant que $[S] = [T]$. Nous avons $P \in [S]$ ssi $\forall \sigma \prec P\ \sigma \in S$ ssi $\forall \sigma \prec P\ \forall \tau \preceq \sigma\ \tau \in T_{|\sigma|}$ ssi $\forall \tau \prec P\ \forall n \geqslant |\tau|\ \tau \in T_n$ ssi $\forall \sigma \prec P\ \sigma \in T$ ssi $P \in [T]$. ∎

Ce chapitre porte principalement sur l'étude des classes d'ensembles correspondant aux chemins d'arbres calculables. Nous verrons avec la proposition 3.5 qu'il existe des arbres calculables infinis ne contenant aucun chemin calculable infini, puis nous déterminerons la puissance de calcul exacte qui est nécessaire au calcul d'un chemin dans n'importe quel arbre calculable infini. Cette étude constitue une des briques de base des mathématiques à rebours, que nous verrons dans la partie III.

## 2. Topologie sur l'espace de Cantor

La topologie est une branche des mathématiques qui formalise de manière abstraite les notions de limite et de continuité, et qui par extension étudie les propriétés d'objets géométriques invariantes par déformation continue. Le lecteur qui n'a jamais étudié cette branche peut se rassurer : nous n'avons besoin pour le développement des chapitres à venir que d'éléments très basiques de cette théorie, que nous présentons ici.

**Notation**

On notera $i^\infty$ la suite infinie qui répète le bit $i \in \{0, 1\}$.

### 2.1. Ouverts et fermés

Comme nous l'avons déjà mentionné, l'espace de Cantor $2^{\mathbb{N}}$ est similaire à l'ensemble des réels de l'intervalle $[0, 1]$. Un élément $X \in 2^{\mathbb{N}}$ peut aussi être vu comme le développement binaire du réel $0.X(0)X(1)X(2)\ldots$, avec toutefois une différence subtile : les éléments $\sigma 10^\infty$ et $\sigma 01^\infty$ sont deux éléments distincts de $2^{\mathbb{N}}$ mais correspondent au même réel. Cette différence mise à part, on peut voir $2^{\mathbb{N}}$ comme un intervalle, et les sous-ensembles les plus simples de $2^{\mathbb{N}}$ seront simplement les intervalles de la forme $[\sigma]$, que l'on appellera aussi *cylindre*.

**Notation**

Étant donné une chaîne $\sigma \in 2^{<\mathbb{N}}$, on écrit $[\sigma]$ pour l'ensemble

$$\{X \in 2^{\mathbb{N}} \ : \ X \succeq \sigma\}.$$

On appellera *cylindre* un ensemble de la forme $[\sigma]$.

Étant donné une chaîne $\sigma \in 2^{<\mathbb{N}}$, on peut voir $[\sigma]$ comme un intervalle de suites binaires infinies : celles qui sont lexicographiquement comprises entre $\sigma 0^\infty$ et $\sigma 1^\infty$. Avec cette vision en tête, les classes dites *ouvertes* du Cantor sont simplement les réunions quelconques d'intervalles.

**Définition 2.1.** Les classes *ouvertes* du Cantor $2^{\mathbb{N}}$ sont les réunions quelconques de cylindres, c'est-à-dire les ensembles de la forme $\bigcup_{\sigma \in W} [\sigma]$ pour un ensemble $W \subseteq 2^{<\mathbb{N}}$. Les classes *fermées* sont les complémentaires des classes ouvertes. ♢

Le lecteur non habitué à ces concepts pourra, pour se faire la main, démontrer sur les définitions que les réunions finies de cylindres sont à la fois des ouverts et des fermés.

**Notation**

Étant donné un ensemble $W \subseteq 2^{<\mathbb{N}}$, on écrira $[W]$ pour dénoter son ouvert correspondant, c'est-à-dire la classe $\bigcup_{\sigma \in W} [\sigma]$.

Il est clair d'après la définition que les ouverts sont clos par réunion quelconque, et donc par passage au complémentaire que les fermés sont clos par intersection quelconque. Nous introduisons ci-après un élément de vocabulaire qui reviendra souvent dans la manipulation des ouverts et des fermés.

**Notation**

Étant donné une réunion dénombrable d'ensembles $\bigcup_n \mathcal{B}_n$, on dira que la réunion est *croissante* si $\mathcal{B}_n \subseteq \mathcal{B}_{n+1}$ pour tout $n$. De la même manière, on dira qu'une intersection $\bigcap_n \mathcal{B}_n$ est *décroissante* si $\mathcal{B}_{n+1} \subseteq \mathcal{B}_n$ pour tout $n$.

La représentation mentale d'un ensemble ouvert devrait être à peu près claire pour le lecteur : les réunions de briques simples que sont les cylindres. Voici un exemple illustratif.

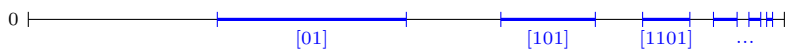



FIGURE 2.2 – *Illustration de l'ouvert* $[01] \cup [101] \cup [1101] \cup [11101] \cup \ldots$

Nous montrons à présent que l'on peut considérer sans perte de généralité que les intersections d'ouverts sont décroissantes et les réunions de fermés croissantes (en particulier, car $\bigcap_n \mathcal{U}_n = \bigcap_n (\bigcap_{m \leqslant n} \mathcal{U}_m)$) :

**Proposition 2.3.** Une intersection finie d'ouverts est un ouvert. Par passage au complémentaire, une réunion finie de fermés est fermée. ⋆

PREUVE. Soient $\mathcal{U}_0, \mathcal{U}_1 \subseteq 2^{\mathbb{N}}$ deux classes ouvertes. Un ensemble $X$ appartient à $\mathcal{U}_0 \cap \mathcal{U}_1$ ssi il appartient à un cylindre $[\sigma_0] \subseteq \mathcal{U}_0$ ainsi qu'à un cylindre $[\sigma_1] \subseteq \mathcal{U}_1$. Donc, $\mathcal{U}_0 \cap \mathcal{U}_1 = \bigcup_{[\sigma_0] \subseteq \mathcal{U}_0, [\sigma_1] \subseteq \mathcal{U}_1} [\sigma_0] \cap [\sigma_1]$. ∎

Les ensembles fermés s'avèrent plus délicats à décrire. Cela n'est pas forcément surprenant. En guise d'analogie, disons que l'on peut bien connaître son quartier, sans avoir une idée précise du reste du monde. La prise de conscience de cette complexité et la manière de l'appréhender figurent déjà dans les travaux de Cantor, par exemple au travers du fameux théorème de Cantor-Bendixson. Il existe toutefois une manière simple de représenter les fermés de $2^{\mathbb{N}}$ : en tant que chemins infinis d'un arbre.

**Proposition 2.4.** Une classe $\mathcal{P} \subseteq 2^{\mathbb{N}}$ est fermée si, et seulement si, il existe un arbre $T \subseteq 2^{<\mathbb{N}}$ tel que $\mathcal{P} = [T]$. ⋆

Preuve. Soit $\mathcal{P}$ une classe fermée, et soit $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$ son complémentaire, avec $W \subseteq 2^{<\mathbb{N}}$. On définit l'arbre $T \subseteq 2^{<\mathbb{N}}$ comme étant l'ensemble des chaînes $\sigma$ n'ayant aucun préfixe dans $W$. Par définition, $T$ est clos par préfixe, et est donc un arbre. Montrons $[T] = \mathcal{P}$. On a $X \in [T]$ ssi aucun préfixe $\sigma \prec X$ n'est dans $W$ ssi $X \notin \bigcup_{\sigma \in W} [\sigma]$ ssi $X \in \mathcal{P}$. Donc, $[T] = \mathcal{P}$.

Inversement, si $T \subseteq 2^{<\mathbb{N}}$ est un arbre, la classe $[T]$ de ses chemins est un fermé, car elle est le complémentaire de la classe $\bigcup_{\sigma \notin T} [\sigma]$. ∎

À titre d'exemple, le lecteur peut consulter la figure 2.5, où se trouve l'arbre qui représente le complémentaire de l'ouvert décrit dans la figure 2.2.

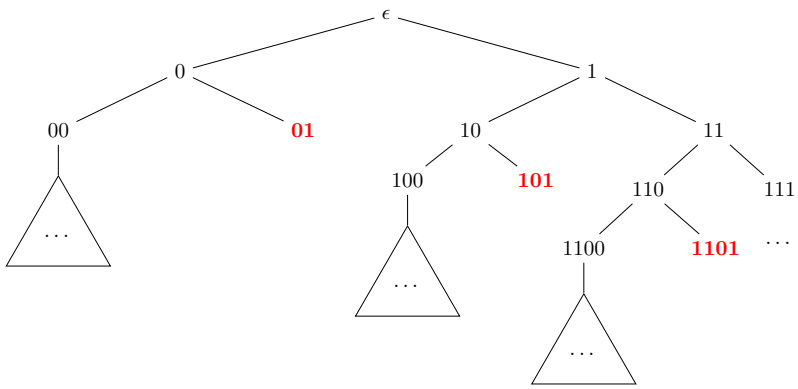



Figure 2.5 – *Illustration de l'arbre représentant le complémentaire dans* $2^{\mathbb{N}}$ *de l'ouvert*

$$[01] \cup [101] \cup [1101] \cup [11101] \cup \ldots$$

*Les nœuds en gras correspondent aux cylindres constituant les briques de base de l'ouvert. Les triangles représentent un sous-arbre « plein » à partir du nœud où ils se trouvent.*

### 2.2. Compacité

La *compacité* est une notion fondamentale de topologie. Elle est généralement définie via la propriété de Borel-Lebesgue, qui dans l'espace de Cantor se formule comme suit.

**Définition 2.6.** Une classe $\mathcal{P} \subseteq 2^{\mathbb{N}}$ a la *propriété de Borel-Lebesgue*[a] si pour toute collection d'ouverts $(\mathcal{O}_n)_{n \in \mathbb{N}}$ telle que $\mathcal{P} \subseteq \bigcup_n \mathcal{O}_n$, il existe un ensemble fini $F \subseteq \mathbb{N}$ tel que $\mathcal{P} \subseteq \bigcup_{n \in F} \mathcal{O}_n$. On dira qu'une classe

possédant la propriété de Borel-Lebesgue est *compacte*. ♢

*a.* Appelée aussi ≪ propriété de Heine-Borel ≫.

Nous montrons avec la proposition suivante que dans l'espace de Cantor, les classes compactes sont exactement les fermés, et le lecteur pourra constater en lisant la preuve que le lemme de König faible peut être vu comme une reformulation du fait que les classes fermées sont compactes.

**Proposition 2.7.** Une classe de $2^{\mathbb{N}}$ est fermée si, et seulement si, elle a la propriété de Borel-Lebesgue. ⋆

Preuve. Soit $\mathcal{P} \subseteq 2^{\mathbb{N}}$ un fermé, et soit $(\mathcal{O}_n)_{n\in\mathbb{N}}$ une collection d'ouverts telle que $\mathcal{P} \subseteq \bigcup_n \mathcal{O}_n$, et soit $T \subseteq 2^{<\mathbb{N}}$ un arbre tel que $[T] = \mathcal{P}$. Soit $S \subseteq 2^{<\mathbb{N}}$ l'arbre des chaînes $\sigma \in T$ telles que $[\sigma] \not\subseteq \bigcup_{n<|\sigma|} \mathcal{O}_n$. Si l'arbre $S$ est fini, il existe une longueur $\ell$ telle que pour tout $\sigma \in T$ tel que $|\sigma| = \ell$,

$$[\sigma] \subseteq \bigcup_{n<\ell} \mathcal{O}_n.$$

Il s'ensuit que $[T] \subseteq \bigcup_{\sigma\in T,\, |\sigma|=\ell} [\sigma] \subseteq \bigcup_{n<\ell} \mathcal{O}_n$. Si $S$ est infini, par le lemme faible de König, $[S] \neq \emptyset$. Soit $P \in [S]$. Montrons que $P \notin \bigcup_n \mathcal{O}_n$ pour en déduire une contradiction, car $[S] \subseteq [T] \subseteq \bigcup_n \mathcal{O}_n$. Soit $i \in \mathbb{N}$. Par définition de $[S]$, pour tout $\ell$, on a $P\restriction_\ell \in S$, et donc par définition de $S$ on a $[P\restriction_\ell] \not\subseteq \bigcup_{n<\ell} \mathcal{O}_n$ ; en particulier, pour tout $\ell > i$, $[P\restriction_\ell] \not\subseteq \mathcal{O}_i$. Comme $\mathcal{O}_i$ est un ouvert, il s'ensuit que $P \notin \mathcal{O}_i$.

Supposons à présent qu'une classe $\mathcal{B}$ admette la propriété de Borel-Lebesgue. Pour tout $X \notin \mathcal{B}$, soit $\mathcal{O}_X$ l'ouvert correspondant au complémentaire de la classe $\{X\}$ (il s'agit de la réunion des cylindres $[\sigma i]$ pour toute chaîne $\sigma$ et tout $i$ tel que $X(|\sigma|) \neq i$). En particulier, on a $\mathcal{B} = \bigcap_{X\notin\mathcal{B}} \mathcal{O}_X$. Notons que chaque $\mathcal{O}_X$ est une réunion de cylindres et que chaque cylindre est ouvert. Donc, par la propriété de Borel-Lebesgue, on peut trouver pour tout $X$ un ensemble fini de cylindres $F_X$ tel que $\mathcal{B} \subseteq \bigcup_{\sigma\in F_X} [\sigma] \subseteq \mathcal{O}_X$. En particulier, $\mathcal{B} = \bigcap_{X\notin\mathcal{B}} \bigcup_{\sigma\in F_X} [\sigma]$. Chaque réunion $\bigcup_{\sigma\in F_X} [\sigma]$ est une classe fermée en tant que réunion finie de cylindres. Comme une intersection arbitraire de fermés est une classe fermée, on en déduit que $\mathcal{B}$ est une classe fermée. ■

En pratique, nous utiliserons la conséquence suivante de la compacité : toute intersection dénombrable et décroissante de fermés non vides, est non vide, ce que nous démontrons ici.

**Proposition 2.8.** Soit $\mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \dots$ une suite décroissante de fermés non vides. Alors $\bigcap_n \mathcal{P}_n$ est non vide. ⋆

PREUVE. Soit $T_n$ un arbre tel que $[T_n] = \mathcal{P}_n$. On peut supposer, sans perte de généralité, que $T_{n+1} \subseteq T_n$. Montrons que $\bigcap_n [T_n] = [\bigcap_n T_n]$. On a $X \in \bigcap_n [T_n]$ ssi $X\restriction_m \in T_n$ pour tout $m, n$, ou encore ssi $X \in [\bigcap_n T_n]$. Donc, $\bigcap_n [T_n] = [\bigcap_n T_n]$. Soit $T = \bigcap_n T_n$. En particulier, $[T] = \bigcap_n \mathcal{P}_n$.

Supposons par l'absurde que $[T] = \bigcap_n \mathcal{P}_n$ est vide. D'après le lemme de König, il y a donc un entier $a$ tel qu'aucune chaîne $\sigma$ de taille supérieure ou égale à $a$ n'est dans $T$. Comme les chaînes de taille $a$ sont en quantité finie et que la suite $(T_n)_{n\in\mathbb{N}}$ est décroissante par l'inclusion, il doit donc y avoir entier $m$ tel qu'aucune de ces chaînes n'appartient à $T_m$. Ainsi, $[T_m]$ est vide, ce qui contredit les hypothèses. ∎

### 2.3. Continuité

Abordons une autre notion topologique que nous mentionnerons ça et là dans les chapitres à venir. La continuité, autre notion centrale de topologie, se cache de manière inattendue en calculabilité sous l'idée suivante.

Une fonctionnelle Turing $\Phi$ peut être vue comme une fonction partielle de $2^{\mathbb{N}}$ dans $2^{\mathbb{N}}$, dont l'entrée est un oracle $X$, et le résultat, que nous notons $\Phi^X$ ou $\Phi(X)$, est l'ensemble $Y$ tel que $\Phi^X(n)\downarrow = Y(n)$. Cette fonction de $2^{\mathbb{N}}$ vers $2^{\mathbb{N}}$ n'est bien entendu définie que pour les oracles $X$ tels que $\forall n\ \Phi^X(n)\downarrow \in \{0,1\}$.

Nous avons vu que quand $\Phi^X(n)\downarrow = v$, par la propriété de l'usage (voir la définition 4-4.2), seul un segment initial fini $\sigma$ de l'oracle $X$ est utilisé. Plus généralement, si $\Phi^X \succeq \tau$ (ce qui signifie $\forall n < |\tau|\ \Phi^X(n)\downarrow = \tau(n)$), seule une partie finie $\sigma$ de l'oracle est utilisée pour s'en rendre compte. Il s'ensuit que pour tout $X \in [\sigma]$, $\Phi^X \succeq \tau$, et donc $\{\Phi^X : X \in [\sigma]\} \subseteq [\tau]$.

Le lecteur ayant suivi un cours d'introduction à la topologie reconnaîtra dans cette idée la notion de continuité : pour tout ouvert de l'espace d'arrivée aussi « petit » que l'on veut — en pratique un cylindre $[\tau]$, il existe un ouvert de l'espace de départ suffisamment « petit » — en pratique un cylindre $[\sigma]$ — tel que tout $X \in [\sigma]$ est envoyé à l'intérieur de $[\tau]$ : concrètement, la chaîne $\sigma$ est « envoyée » vers la chaîne $\tau$.

**Définition 2.9.** Une fonction (éventuellement partielle) $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ est *continue en* $X \in \operatorname{dom} f$ si pour tout cylindre $[\tau]$ contenant $f(X)$, il existe un cylindre $[\sigma]$ contenant $X$ tel que $f([\sigma]) \subseteq [\tau]$. On dira qu'une fonction est *continue* si elle est continue en $X$ pour tout $X \in \operatorname{dom} f$. ♢

On considérera en général des fonctions continues sur tout leur domaine de définition, qui admettent alors une caractérisation équivalente en termes de pré-image d'ouverts.

**Proposition 2.10.** Une fonction (éventuellement partielle) $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ est continue si, et seulement si, il existe pour tout ouvert $\mathcal{U} \subseteq 2^{\mathbb{N}}$ un ouvert $\mathcal{V} \subseteq 2^{\mathbb{N}}$ tel que $f^{-1}(\mathcal{U}) = \mathcal{V} \cap \operatorname{dom} f$. Si la fonction est totale, on a alors $f^{-1}(\mathcal{U})$ ouvert pour tout ouvert $\mathcal{U}$. ⋆

Preuve. Soient $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ une fonction continue et $\mathcal{U} \subseteq 2^{\mathbb{N}}$ un ouvert. Comme $\mathcal{U}$ est ouvert, pour tout $X \in f^{-1}(\mathcal{U})$ il existe un cylindre $[\tau_X]$ contenant $f(X)$ tel que $[\tau_X] \subseteq \mathcal{U}$. Par continuité de $f$, pour tout $X$, il existe un cylindre $[\sigma_X]$ contenant $X$ tel que $f([\sigma_X]) \subseteq [\tau_X]$. Alors,

$$\operatorname{dom} f \cap \bigcup_{X \in f^{-1}(\mathcal{U})} [\sigma_X] = f^{-1}(\mathcal{U}).$$

Réciproquement, supposons que pour tout ouvert $\mathcal{U} \subseteq 2^{\mathbb{N}}$, il existe un ouvert $\mathcal{V}$ tel que $\operatorname{dom} f \cap \mathcal{V} = f^{-1}(\mathcal{U})$. Soit $Y \in \operatorname{Im} f$, et soit $[\tau]$ un cylindre contenant $Y$. Soit un ouvert $\mathcal{V}$ tel que $\operatorname{dom} f \cap \mathcal{V} = f^{-1}([\tau])$. Comme $\mathcal{V}$ est un ouvert, il existe $W \subseteq 2^{<\mathbb{N}}$ tel que $\bigcup_{\sigma \in W} [\sigma] = \mathcal{V}$. Notons que $W \neq \emptyset$, car $Y \in \operatorname{Im} f \cap [\tau]$, donc il existe un cylindre $\sigma \in W$. On a

$$f([\sigma]) = f(\operatorname{dom} f \cap [\sigma]) \subseteq [\tau].$$ ∎

Une fonctionnelle calculable $\Phi$ est donc toujours aussi une fonction continue sur son domaine de définition, c'est-à-dire sur l'espace des $X$ tels que $\Phi(X, n) \downarrow \in \{0, 1\}$ pour tout $n$. En revanche, une fonction continue n'a *a priori* aucune raison d'être calculable : il se peut que n'importe quel morceau fini de la sortie de la fonction puisse être déterminé par un morceau fini de l'entrée, mais que ce « déterminisme » ne soit pas calculable. À titre d'exemple, considérons la fonction $\Phi$ qui sur n'importe quel ensemble $X$ associe $X \oplus \emptyset'$. Une telle fonction $\Phi$ est continue, mais pas calculable. Elle est en revanche calculable avec l'aide de $\emptyset'$.

En calculabilité, la fonction non continue par excellence est celle qui à $X$ associe $X'$ : en effet, pour savoir si $n \in X'$, il faut savoir si $\Phi_n(X, n) \downarrow$, et pour cela potentiellement connaître une infinité de bits de $X$ (en particulier, si $\Phi_n(X, n) \uparrow$). Nous verrons des versions effectives de certains théorèmes bien connus d'analyse, qui stipulent que toute fonction non continue, mais pas trop complexe, par exemple $X \mapsto X'$, est malgré tout continue sur un « large » ensemble de points, en particulier sur une classe co-maigre (voir le théorème 10-3.20) et sur une classe de mesure arbitrairement grande (voir le théorème 19-3.8).

### 2.4. Classes parfaites

Une dernière notion topologique que nous utiliserons est celle de classes parfaites. Ce sont les classes qui sont l'image d'une injection continue

de $2^{\mathbb{N}}$ vers $2^{\mathbb{N}}$, c'est-à-dire exactement les classes de la forme $[T]$ pour un f-arbre $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ (voir la section 7-5). Ces classes-là sont donc toujours fermées et peuvent se représenter par un arbre. Par extension, on parlera donc aussi d'arbre parfait.

**Définition 2.11.** Un arbre non vide $T \subseteq 2^{<\mathbb{N}}$ est parfait si tout nœud de $T$ a deux extensions incompatibles dans $T$. ♢

Nous avons vu avec l'exercice 1.7 qu'étant donné un fermé $\mathcal{F}$ représenté par un arbre $T$, on peut considérer sans perte de généralité — si l'on ne s'occupe pas de l'effectivité — que $T$ ne contient que des nœuds extensibles. En revanche, un nœud extensible n'a pas nécessairement deux extensions incompatibles dans le cas général. Quand cela arrive, cela signifie qu'il y a exactement un chemin infini passant par ce nœud. On appelle de tels chemins des *points isolés*. Pour une classe $\mathcal{A}$ arbitraire, la définition correspondante est la suivante.

**Définition 2.12.** Soit $\mathcal{A} \subseteq 2^{\mathbb{N}}$. Un élément $X \in \mathcal{A}$ est un *point isolé* s'il existe un préfixe $\sigma \prec X$ tel que $[\sigma] \cap \mathcal{A} = \{X\}$. ♢

La définition usuelle de classe parfaite découle alors de celle de point isolé.

**Définition 2.13.** Une classe non vide $\mathcal{F} \subseteq 2^{\mathbb{N}}$ est *parfaite* si elle est fermée et n'a pas de point isolé. De manière équivalente, $\mathcal{F} = [T]$ pour un arbre parfait $T \subseteq 2^{<\mathbb{N}}$. ♢

Les classes parfaites sont d'une grande importance, notamment car elles permettent de construire des arguments de cardinalité : toute classe parfaite est par définition en bijection continue avec $2^{\mathbb{N}}$, et a donc la même cardinalité que $2^{\mathbb{N}}$. Par ailleurs, si une classe $\mathcal{A} \subseteq 2^{\mathbb{N}}$ contient une classe parfaite, alors on a une injection de $2^{\mathbb{N}}$ dans $\mathcal{A}$. L'injection identité de $\mathcal{A}$ dans $2^{\mathbb{N}}$ nous donne alors $|\mathcal{A}| = |2^{\mathbb{N}}|$. Il s'agit en fait grosso modo *de la seule manière* de montrer qu'une classe $\mathcal{A} \subseteq 2^{\mathbb{N}}$ a la puissance du continu. Nous en reparlerons dans la section 9-4, ainsi que dans la section 30-4.

Voici pour terminer une application simple de la notion de classe parfaite à l'étude de la cardinalité.

**Proposition 2.14.** Toute classe fermée dénombrable et non vide $\mathcal{F}$ contient des points isolés. On peut injecter $2^{\mathbb{N}}$ dans toute classe fermée non vide ne contenant pas de point isolé. ⋆

Preuve. Supposons que $\mathcal{F}$ ne contienne pas de point isolé. Soit $\mathcal{F} = [T]$ pour un arbre $T$ n'ayant que des nœuds extensibles. Comme $\mathcal{F}$ ne contient pas de point isolé, tous les nœuds de $T$ ont alors deux extensions incompatibles. On a alors une injection de $2^{\mathbb{N}}$ dans $\mathcal{F}$. Si une classe fermée $\mathcal{F}$

est dénombrable, on ne peut injecter $2^{\mathbb{N}}$ dans $\mathcal{F}$, et elle contient donc par contraposée des points isolés. ∎

**Corollaire 2.15 (Cantor)**
*L'hypothèse du continu est vraie pour les classes fermées de $2^{\mathbb{N}}$ : elles sont soit finies, soit dénombrables, soit de cardinalité $|2^{\mathbb{N}}|$.*

Preuve. Soit $\mathcal{F}$ un fermé, et soit $W \subseteq 2^{<\mathbb{N}}$ l'ensemble des chaînes $\sigma$ telles que $\mathcal{F} \cap [\sigma]$ est fini ou dénombrable. Soit $\mathcal{F}' = \mathcal{F} \setminus \bigcup_{\sigma \in W} [\sigma]$. Notons que $\mathcal{F}' \subseteq \mathcal{F}$ est toujours une classe fermée. Si $\mathcal{F}'$ est vide, alors d'après le lemme de König (ou la compacité) il n'est besoin d'enlever à $\mathcal{F}$ qu'un nombre fini de chaînes $\sigma_0, \ldots, \sigma_n \in W$ pour que $\mathcal{F}' = \mathcal{F} \setminus ([\sigma_0] \cup \cdots \cup [\sigma_n])$ soit vide. Comme chaque classe $\mathcal{F} \cap [\sigma_i]$ est dénombrable, on a vidé $\mathcal{F}$ en lui enlevant une quantité dénombrable de points. Donc, $\mathcal{F}$ est dénombrable. Sinon, $\mathcal{F}'$ n'est pas vide, et en plus de cela il ne peut contenir de points isolés (car si $\mathcal{F}' \cap [\sigma]$ ne contient qu'un seul élément, alors $\mathcal{F} \cap [\sigma]$ est dénombrable). Par la proposition 2.14, on a une injection de $2^{\mathbb{N}}$ vers $\mathcal{F}'$. ∎

Ce résultat sera étendu avec le corollaire 30-3.3.

# 3. Classes $\Pi^0_1$

Nous nous intéressons à présent à la version *effective* des classes ouvertes et fermées. On emploie souvent le terme *effectif* plutôt que calculable pour ce genre d'objets, car ils ne sont pas nécessairement calculables dans le sens où l'on peut précisément en connaître les moindres détails, mais ils admettent malgré tout une certaine description tangible, effective, fournie par un algorithme.

**Définition 3.1.** Une classe $\mathcal{U} \subseteq 2^{\mathbb{N}}$ est dite $\Sigma^0_1$ s'il existe un ensemble c. e. $W \subseteq 2^{<\mathbb{N}}$ tel que $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$. Une classe $\mathcal{P} \subseteq 2^{\mathbb{N}}$ est dite $\Pi^0_1$ si son complémentaire est une classe $\Sigma^0_1$. ♢

Les $\Sigma^0_1$ et $\Pi^0_1$ sont respectivement les ouverts et fermés *effectifs* de l'espace de Cantor. On appellera *code* d'une classe $\Sigma^0_1$ $\mathcal{U}$ un entier $e$ tel que $\mathcal{U} = \bigcup_{\sigma \in W_e} [\sigma]$. De même, le *code* d'une classe $\Pi^0_1$ $\mathcal{P}$ est le code de la classe $\Sigma^0_1$ $\mathcal{U} = 2^{\mathbb{N}} \setminus \mathcal{P}$. Cela nous permet de parler de calculabilité uniforme sur les suites de classes $\Sigma^0_1$ ou $\Pi^0_1$ en considérant la calculabilité de leur suite de codes. Les classes $\Pi^0_1$ sont un exemple important et très étudié en calculabilité.

**Remarque**

Il convient de bien distinguer les ensembles d'entiers $\Sigma^0_1$ et $\Pi^0_1$, qui sont les premiers niveaux de la hiérarchie arithmétique (voir le chapitre 5) des classes $\Sigma^0_1$ et $\Pi^0_1$ qui sont les ouverts et fermés effectifs respectifs de l'espace de Cantor. Il existe cependant des liens entre ces notions.

Commençons par le fait que la proposition 2.3, qui stipule qu'une intersection finie d'ouverts est un ouvert, fonctionne également avec les classes $\Sigma^0_1$.

**Proposition 3.2.** Les classes $\Sigma^0_1$ sont closes par intersection finie. Par passage au complémentaire, les classes $\Pi^0_1$ sont closes par réunion finie. ⋆

Preuve. Soient $\mathcal{U}_0 = \bigcup_{\sigma\in W_0}[\sigma]$ et $\mathcal{U}_1 = \bigcup_{\sigma\in W_1}[\sigma]$ deux classes $\Sigma^0_1$. Alors, l'ouvert $\mathcal{U}_0 \cap \mathcal{U}_1$ est décrit par l'ensemble c. e. qui énumère la plus longue chaîne parmi $\sigma_0, \sigma_1$ pour tous $\sigma_0 \in W_0$ et $\sigma_1 \in W_1$ telles que $\sigma_0 \preceq \sigma_1$ ou telles que $\sigma_1 \preceq \sigma_0$. La chaîne ainsi énumérée correspond à $[\sigma_0] \cap [\sigma_1]$. ∎

La première étape pour mieux appréhender la nature des classes $\Pi^0_1$ est sans doute de prouver la version effective du théorème 2.4 : les $\Pi^0_1$ sont exactement les chemins infinis d'arbres calculables.

**Proposition 3.3.** Une classe $\mathcal{P}$ est $\Pi^0_1$ si, et seulement si, il existe un arbre calculable $T \subseteq 2^{<\mathbb{N}}$ tel que $[T] = \mathcal{P}$. ⋆

Preuve. Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre calculable. La classe $[T] = \{X : \forall n\, X\restriction_n \in T\}$ est alors bien $\Pi^0_1$. En effet, son complémentaire est la classe $\Sigma^0_1$ décrite par la réunion des cylindres $[\sigma]$ tels que $\sigma \notin T$.

Supposons que $\mathcal{P}$ est $\Pi^0_1$, et soit $\mathcal{U} = \bigcup_{\sigma\in W}[\sigma]$ son complémentaire. On calcule l'arbre $T \subseteq 2^{<\mathbb{N}}$ suivant : à l'étape de calcul $t$, pour toute chaîne $\sigma \in 2^{<\mathbb{N}}$ de taille $t$, on décide $\sigma \in T$ ssi pour tout préfixe $\tau \preceq \sigma$ on a $\tau \notin W[t]$.

Il est clair que $T$ est clos par préfixe : si $\sigma$ de taille $t$ est dans $T$, alors aucun préfixe de $\sigma$ n'est dans $W$ à l'étape de calcul $t$, donc pour $s \leqslant t$, aussi aucun préfixe de $\sigma\restriction_s$ n'est dans $W$ à l'étape de calcul $s$. À présent, si $X \in \mathcal{P}$, alors aucun préfixe $\sigma$ de $X$ n'est dans $W$, et chacun de ces préfixes sera donc dans $T$. À l'inverse, si $X \notin \mathcal{P}$, alors un préfixe $\sigma$ de $X$ rentre dans $W$ à une certaine étape $t$. Par construction, aucune chaîne $\tau \succeq \sigma$ de taille supérieure à $t$ ne sera dans $T$. Donc, $\mathcal{P} = [T]$. ∎

Le code d'un arbre calculable $T \subseteq 2^{<\mathbb{N}}$ est un entier $e$ tel que $\Phi_e = T$. Notons que la preuve de la proposition 3.3 est uniforme, et permet de passer calculatoirement d'un code d'une classe $\Pi^0_1$ à un code de l'arbre correspondant, et inversement. On pourra donc considérer indistinctement le code des classes $\Pi^0_1$ et des arbres calculables dans les preuves à venir. La

proposition suivante établit un lien avec les classes $\Sigma^0_1$ et $\Pi^0_1$ et la hiérarchie arithmétique.

**Proposition 3.4.** Soit $\mathcal{P} \subseteq 2^{\mathbb{N}}$ une classe.

(1) La classe $\mathcal{P}$ est $\Sigma^0_1$ si, et seulement si, $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \exists n\ R(X\restriction_n)\}$ pour un prédicat calculable $R \subseteq 2^{<\mathbb{N}}$.

(2) La classe $\mathcal{P}$ est $\Pi^0_1$ si, et seulement si, $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall n\ R(X\restriction_n)\}$ pour un prédicat calculable $R \subseteq 2^{<\mathbb{N}}$. ⋆

Preuve. Pour (2), étant donné $\mathcal{P}$ une classe $\Pi^0_1$, il suffit de considérer l'arbre calculable $T$ tel que $[T] = \mathcal{P}$. Le prédicat calculable est simplement $T$. À l'inverse, si $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall n\ R(X\restriction_n)\}$, alors l'arbre calculable donné par $\sigma \in T$ ssi $\forall \tau \preceq \sigma\ R(\tau)$ est tel que $[T] = \mathcal{P}$.

On obtient (1) par passage au complémentaire. ∎

Voyons à présent quelques généralités, en premier lieu la preuve que le lemme de König ne relève pas des mathématiques calculables : certaines classes $\Pi^0_1$ non vides — et donc certains arbres calculables infinis — ne contiennent aucun point calculable. Nous verrons tout au long des chapitres à venir de nombreux exemples de classes $\Pi^0_1$ ne contenant aucun point calculable. Nous anticipons en particulier pour la proposition suivante sur l'exemple le plus simple à définir : la classe des ensembles $\mathrm{DNC}_2$ de la proposition 6.4.

**Proposition 3.5.** Il existe des classes $\Pi^0_1$ non vides ne contenant aucun ensemble calculable. ⋆

Preuve. On définit la classe

$$\mathcal{P} = \{X \in 2^{\mathbb{N}} \ :\ \forall e\ \forall t\ \Phi_e(e)[t]\uparrow \vee\ \Phi_e(e)[t]\downarrow \neq X(e)\}.$$

La classe $\mathcal{P}$ contient tous les ensembles $X$ tels que $X(n)$ peut prendre n'importe quelle valeur si $\Phi_n(n)\uparrow$, et qui sont toujours différents de $\Phi_n(n)$ si $\Phi_n(n)\downarrow$. Il est clair que cette classe est non vide (l'arrêt des programmes informatique calcule par exemple facilement un élément de $\mathcal{P}$). Par la proposition 3.4, $\mathcal{P}$ est une classe $\Pi^0_1$. De plus, la classe $\mathcal{P}$ ne contient aucun ensemble calculable : si $X$ est calculable, alors il doit exister $e$ tel que $\Phi_e(e)\downarrow = X(e)$. ∎

Continuons à présent sur un autre aspect clef : les exemples de classes $\Pi^0_1$ non vides ne contenant aucun point calculable sont nécessairement indénombrables. Ils ne peuvent pas contenir de *points isolés*, c'est-à-dire des ensembles $X$ tels que pour un certain $n$, aucun autre ensemble que $X$ et étendant $X\restriction_n$ n'appartient au $\Pi^0_1$.

**Proposition 3.6.** Si $\mathcal{P}$ est une classe $\Pi^0_1$ contenant exactement un élément $X$, alors, $X$ est calculable. ⋆

Preuve. Soit $T \subseteq 2^{<\mathbb{N}}$ l'arbre calculable tel que $[T] = \mathcal{P}$. Soit l'algorithme suivant : on cherche le plus petit $t$ tel que soit pour toute chaîne $\sigma \succeq 0$ de taille $t$ on a $\sigma \notin T$, soit pour toute chaîne $\sigma \succeq 1$ de taille $t$ on a $\sigma \notin T$. Notez qu'exactement un des deux événements doit nécessairement arriver : si les deux événements arrivent, la classe est vide. Si aucun des deux n'arrive, il y a une infinité de chaînes dans $T$ qui étendent 0 et aussi une infinité qui étendent 1. D'après le lemme de König, $T$ contient donc au moins deux chemins infinis : un qui étend 0 et un qui étend 1, ce qui contredit les hypothèses sur $\mathcal{P}$.

Une fois un des deux événements arrivés, on sait donc si $X$ commence par 0 ou 1. On voit aisément comment continuer par récurrence : une fois $X\restriction_n$ calculé, on cherche le plus petit $t$ tel que pour toute chaîne $\sigma \succeq X\restriction_n 0$ de taille $t$ on a $\sigma \notin T$, ou pour toute chaîne $\sigma \succeq X\restriction_n 1$ de taille $t$ on a $\sigma \notin T$. Une fois l'un des deux événements arrivé, la valeur de $X(n)$ est connue. ∎

**Corollaire 3.7**
*Les points isolés de toute classe $\Pi^0_1$ sont calculables.*

Preuve. Soit $\mathcal{P}$ une classe $\Pi^0_1$, et soit $X \in \mathcal{P}$ un point isolé. Par définition, on a un préfixe $\sigma \prec X$ tel que $[\sigma] \cap \mathcal{P} = \{X\}$. En particulier, $[\sigma] \cap \mathcal{P}$ est une classe $\Pi^0_1$ contenant exactement un élément, et cet élément est donc calculable. ∎

**Corollaire 3.8**
*Toute classe $\Pi^0_1$ dénombrable contient un ensemble calculable.*

Preuve. Par la proposition 2.14 et le corollaire 3.7. ∎

## 4. Théorèmes de base

Les membres d'une classe $\Pi^0_1$ peuvent être de degrés Turing très différents. Par exemple, l'espace de Cantor $2^{\mathbb{N}}$ est une classe $\Pi^0_1$ contenant des ensembles de chaque degré Turing.
Étant donné une classe $\Pi^0_1$ non vide, on s'intéresse principalement au degré de difficulté du calcul de l'un de ses membres.

**Définition 4.1.** Une *base* pour les classes $\Pi^0_1$ est une classe d'ensembles $\mathcal{C}$ telle que toute classe $\Pi^0_1$ non vide contient un élément de $\mathcal{C}$. ♢

Dans cette section, nous allons prouver un certain nombre de théorèmes qui étant donné une propriété de faiblesse $P$ sont de la forme « Toute classe $\Pi^0_1$ non vide contient un membre satisfaisant $P$. » Ces théorèmes sont appelés « théorèmes de bases », car ils établissent que les membres de $P$ forment une base pour les classes $\Pi^0_1$. Inversement, les « théorèmes d'anti-bases » énoncent l'existence d'une classe $\Pi^0_1$ non vide ne contenant aucun membre satisfaisant une propriété de faiblesse. Le tout premier théorème de base est dû à Kreisel [124], et est laissé en exercice.

**Exercice 4.2.** $(\star)$ Montrer que toute classe $\Pi^0_1$ non vide contient un élément $\emptyset'$-calculable. ◇

On peut faire encore mieux que l'exercice 4.2 via le théorème central dit de « base low », qui stipule que toute classe $\Pi^0_1$ non vide contient un ensemble low. Ce théorème a une importance fondamentale en calculabilité et en mathématique à rebours, notamment pour fournir nombre d'exemples et contre-exemples.

**Théorème 4.3 (Jockusch et Soare [104])**
*Toute classe $\Pi^0_1$ non vide contient un ensemble low.*

Preuve. Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide. Nous allons utiliser $\emptyset'$ pour calculer un élément $Z \in \mathcal{P}$, tout en calculant son saut Turing $Z'$. Pour cela, définissons une suite décroissante uniformément $\emptyset'$-calculable de classes $\Pi^0_1$ non vides $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$ comme suit. Soit $\mathcal{P}_0 = \mathcal{P}$ ; supposons $\mathcal{P}_n$ défini, et considérons la classe

$$\mathcal{B}_n = \{X \in 2^{\mathbb{N}} \ : \ \forall t \ \Phi_n(X, n)[t] \uparrow\} \cap \mathcal{P}_n.$$

Notons que $\mathcal{B}_n$ est aussi une classe $\Pi^0_1$, et que le code d'un arbre calculable $T_n$ tel que $\mathcal{B}_n = [T_n]$ est calculable uniformément en $n$.

On pose à $\emptyset'$ la question de savoir si $\mathcal{B}_n$ est vide : d'après le lemme de König, c'est le cas ssi il existe $m$ tel qu'aucune chaîne $\sigma$ de taille $m$ n'appartient à $T_n$, ce qui est bien un événement $\Sigma^0_1$. Si $\emptyset'$ répond positivement, on note $Y(n) = 1$ et l'on définit $\mathcal{P}_{n+1} = \mathcal{P}_n$. Notons que, dans ce cas, tous les éléments $X \in \mathcal{P}_{n+1}$ sont tels que $\Phi_n(X, n) \downarrow$. Dans le cas inverse, on note $Y(n) = 0$ et l'on définit $\mathcal{P}_{n+1} = \mathcal{B}_n$. Notons que, dans ce cas-ci, tous les éléments de $X \in \mathcal{P}_{n+1}$ sont tels que $\Phi_n(X, n) \uparrow$.

Pour chaque $n$, $\mathcal{P}_n$ est un fermé non vide, et $\bigcap_n \mathcal{P}_n$ est donc non vide. Par construction, l'élément $Y$ calculé par $\emptyset'$ correspond au saut Turing de n'importe quel élément de $\bigcap_n \mathcal{P}_n$. ∎

Nous avons déjà vu avec la proposition 4-9.1 l'existence d'ensembles low et non calculable. Nous en avons à présent une preuve alternative en combinant le théorème 4.3 et la proposition 3.5.

Nous nous attelons à présent au deuxième grand théorème de base pour les classes $\Pi^0_1$ : les ensembles calculatoirement dominés. Nous avons besoin pour cela d'un lemme, qui a aussi son intérêt propre.

**Lemme 4.4.** Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide. Supposons qu'une fonctionnelle $\Phi$ est totale sur tous les chemins de $\mathcal{P}$. Alors, on peut définir uniformément en un code de la classe $\mathcal{P}$ une fonction calculable $g$ qui domine $n \mapsto \Phi(X, n)$ pour tout $X \in \mathcal{P}$. ⋆

Preuve. Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre calculable tel que $[T] = \mathcal{P}$. Montrons que pour tout entier $n$ il existe $t$ tel que $\Phi(\sigma, n)[|\sigma|] \downarrow$ pour toute chaîne $\sigma \in T$ de taille $t$. En effet, dans le cas inverse, il existe un entier $n$ tel que l'ensemble $\{\sigma \in T \ : \ \Phi(\sigma, n)[|\sigma|] \uparrow\}$ contient pour tout $t$ une chaîne de taille $t$ et est donc un sous-arbre infini de $T$, qui contient donc par le lemme de König un chemin infini $X$. On a donc $\forall t \ \Phi(X, n)[t] \uparrow$, ce qui contredit la totalité de $\Phi$ sur tous les oracles de $[T]$.

On peut donc calculer la fonction $g$ qui pour $n$ cherche le plus petit $t$ tel que $\Phi(\sigma, n)[t] \downarrow = v_\sigma$ pour toute chaîne $\sigma \in T$ de taille $t$. Une fois $t$ trouvé, on définit $g(n) = \sum_{|\sigma|=t} v_\sigma + 1$. Il est clair que $g$ domine toutes les fonctions calculables via $\Phi$ par un oracle de $[T]$. ∎

Le théorème suivant est connu sous le nom de « théorème de base calculatoirement dominée ». Avec l'existence d'une classe $\Pi^0_1$ non vide ne possédant pas de membre calculable, ce théorème nous fournit une preuve alternative de l'existence d'ensembles calculatoirement dominés non calculables.

**Théorème 4.5 (Jockusch et Soare [104])**
*Toute classe $\Pi^0_1$ non vide contient un ensemble calculatoiremement dominé.*

Preuve. Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide. Nous allons définir une suite infinie décroissante de classes $\Pi^0_1$ non vides $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$ de telle sorte que $\bigcap_n \mathcal{P}_n$ ne contienne que des ensembles calculatoirement dominés. Soit $\mathcal{P}_0 = \mathcal{P}$. Supposons $\mathcal{P}_n$ défini. Soit $\mathcal{B}_{n,m} = \{X \ : \ \Phi_n(X, m) \uparrow\}$. Notons que chaque classe $\mathcal{B}_{n,m}$ est $\Pi^0_1$. Supposons qu'il existe un entier $m$ tel que $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$. Alors, on définit $\mathcal{P}_{n+1} = \mathcal{P}_n \cap \mathcal{B}_{n,m}$. Notons que pour tout $X \in \mathcal{P}_{n+1}$ on a $\Phi_n(X, m) \uparrow$. Supposons à présent que pour tout $m$ on a $\mathcal{P}_n \cap \mathcal{B}_{n,m} = \emptyset$. Cela implique que la fonctionnelle $\Phi_n$ est totale pour tout $X \in \mathcal{P}_n$. On définit alors $\mathcal{P}_{n+1} = \mathcal{P}_n$. D'après le lemme 4.4, il existe

donc une fonction calculable $g : \mathbb{N} \to \mathbb{N}$ qui domine $m \mapsto \Phi_n(X, m)$ pour tout $X \in \mathcal{P}_{n+1}$.

En tant qu'intersection décroissante de fermés non vides, la classe $\bigcap_n \mathcal{P}_n$ est non vide. Soit $X \in \bigcap_n \mathcal{P}_n$. Par construction, pour tout $n$, si $\Phi_n$ est totale sur l'oracle $X$, alors $m \mapsto \Phi_n(X, m)$ est bornée par une fonction calculable. Donc, $X$ est calculatoirement dominé. ∎

Nous voyons à présent un dernier théorème de base dit « d'évitement de cône » : étant donné un ensemble $X$, on appelle *cône supérieur* de $X$ la classe $\mathcal{C}_X = \{Y \in 2^{\mathbb{N}} : Y \geqslant_T X\}$. Jockusch et Soare [104] ont prouvé que, pour chaque ensemble non calculable $X$, la classe $2^{\mathbb{N}} \setminus \mathcal{C}_X$ est une base pour les classes $\Pi^0_1$. En d'autres termes, si $X$ est un ensemble non calculable, toute classe $\Pi^0_1$ non vide possède un élément qui ne calcule pas $X$. La contraposée, plus naturelle, énonce que si un ensemble est calculable par tous les membres d'une classe $\Pi^0_1$ non vide, il est nécessairement calculable. Notons que si une classe $\Pi^0_1$ a un membre calculable, le résultat est évident, et il ne devient intéressant que pour les classes $\Pi^0_1$ non vides qui n'en ont pas. Tout comme pour le théorème de base calculatoirement dominée, nous avons besoin d'un lemme pour régler le cas d'une fonctionnelle fixée.

**Lemme 4.6.** Soient $X$ un ensemble, $\mathcal{P}$ une classe $\Pi^0_1$ non vide et $\Phi$ une fonctionnelle. Si $\Phi^Y = X$ pour tout $Y \in \mathcal{P}$, alors $X$ est calculable. ⋆

Preuve. Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre calculable tel que $[T] = \mathcal{P}$. Supposons que pour tout $Y \in [T]$, $\Phi^Y = X$. Montrons que pour tout $n$, il existe un $t \in \mathbb{N}$ tel que $\Phi(\sigma, n)[|\sigma|] \downarrow = X(n)$ pour toute chaîne $\sigma \in T$ de taille $t$. En effet, dans le cas contraire, l'ensemble $S = \{\sigma \in T : \Phi(\sigma, n)[|\sigma|] \neq X(n)\}$ est un sous-arbre de $T$ qui contient des éléments de chaque longueur, et il existe donc par le lemme de König un chemin $Y \in [S] \subseteq [T]$ tel que $\Phi^Y(n) \neq X(n)$, contredisant ainsi notre hypothèse.

Soit $g : \mathbb{N} \to \mathbb{N}$ la fonction calculable qui sur l'entrée $n$ cherche $t, v_n \in \mathbb{N}$ tels que $\Phi(\sigma, n)[|\sigma|] \downarrow = v_n$ pour toute chaîne $\sigma \in T$ de taille $t$, et renvoie $v_n$. Nous avons montré que cette fonction était totale. De plus, on a nécessairement $v_n = X(n)$ pour tout $n$, car sinon chaque élément de $\mathcal{P}$ calcule autre chose que $X$ sur le bit $n$. Ainsi, $X$ est calculé par $g$, et est donc calculable. ∎

**Théorème 4.7 (Jockusch et Soare [104])**
*Soient $X$ un ensemble non calculable et $\mathcal{P}$ une classe $\Pi^0_1$ non vide. Alors, il existe un élément de $\mathcal{P}$ qui ne calcule pas $X$.*

PREUVE. Nous allons définir une suite infinie décroissante de classes $\Pi^0_1$ non vides $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$ de telle sorte qu'aucun élément de $\bigcap_n \mathcal{P}_n$ ne calcule $X$. Soit $\mathcal{P}_0 = \mathcal{P}$. Supposons $\mathcal{P}_n$ défini. Soit

$$\mathcal{B}_{n,m} = \{Y \ : \ \Phi_n(Y, m)\!\uparrow \vee \ \Phi_n(Y, m) \ \neq \ X(m)\}.$$

Notons que chaque classe $\mathcal{B}_{n,m}$ est $\Pi^0_1$ (pas uniformément, bien sûr, car on ne connaît pas $X$). Montrons qu'il existe $m$ tel que $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$. Si ce n'était pas le cas, on aurait alors $\Phi^Y_n = X$ pour tout $Y \in \mathcal{P}_n$, contredisant dès lors le lemme 4.6. Il existe donc $m$ tel que $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$, et l'on définit alors $\mathcal{P}_{n+1} = \mathcal{P}_n \cap \mathcal{B}_{n,m}$ pour un tel entier $m$, ce qui nous assure $\Phi_n(X, m)\!\uparrow$ ou $\Phi_n(X, m)\!\downarrow \neq X(m)$ pour tout $X \in \mathcal{P}_{n+1}$.

En tant qu'intersection décroissante de fermés non vides, la classe $\bigcap_n \mathcal{P}_n$ est non vide. Soit $Y \in \bigcap_n \mathcal{P}_n$. Par construction, pour tout $n$, $\Phi^Y_n \neq X$, car $Y \in \mathcal{P}_n$. Donc, $X \not\leqslant_T Y$. ∎

Hirschfeldt [87] a donné une élégante preuve alternative du théorème d'évitement de cône, comme simple conséquence du théorème 4.3 de base low et du théorème 4.5 de base calculatoirement dominée.

PREUVE ALTERNATIVE DU THÉORÈME 4.7. Deux cas se présentent.

- ▷ Cas 1. L'ensemble $X$ est $\Delta^0_2$. En particulier, par la proposition 7-4.7, $X$ est hyperimmune. Par le théorème 4.5 de base calculatoirement dominé, la classe $\mathcal{P}$ contient un ensemble calculatoirement dominé $P$. En particulier, $P$ ne calcule pas $X$.
- ▷ Cas 2. L'ensemble $X$ n'est pas $\Delta^0_2$. Par le théorème 4.3 de base low, la classe $\mathcal{P}$ contient un ensemble low $P$, donc $\Delta^0_2$. En particulier, $P$ ne calcule pas $X$.

Dans chacun des cas, $\mathcal{P}$ contient un élément qui ne calcule pas $X$. ∎

Nous verrons dans les chapitres à venir de nombreux autres théorèmes concernant les classes $\Pi^0_1$.

## 5. Bases pour les classes $\Pi^0_1$ parfaites

Nous avons vu que les classes $\Pi^0_1$ sans élément calculable sont nécessairement des classes parfaites. Ces classes admettent des théorèmes de base renforcés, et l'on peut notamment y construire des sous-classes parfaites dont tous les éléments ont une propriété de faiblesse fixée à l'avance. Nous voyons ici un exemple avec les ensembles calculatoirement dominés.

L'idée est de recommencer la preuve de base calculatoirement dominée, mais en dupliquant la construction étape après étape.

**Théorème 5.1**
*Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide ne contenant aucun élément calculable. Il existe une classe parfaite $\mathcal{B} \subseteq \mathcal{P}$ qui ne contient que des ensembles calculatoirement dominés.*

Preuve. Soit $P_\epsilon = \mathcal{P}$. Supposons que pour $n$ et chaque $\sigma \in 2^{<\mathbb{N}}$ de taille $n$ on ait défini des classes $\Pi^0_1$ deux à deux disjointes et non vides $\mathcal{P}_\sigma \subseteq \mathcal{P}$. On répète la construction du théorème 4.5 pour définir pour chaque $\sigma$ une classe $\Pi^0_1$ non vide $\mathcal{Q}_\sigma \subseteq \mathcal{P}_\sigma$ telle que soit l'on a un $m$ tel que $\Phi_n(X, m)\uparrow$ pour tout $X \in \mathcal{Q}_\sigma$, soit l'on a une fonction calculable $g : \mathbb{N} \to \mathbb{N}$ telle que $\Phi_n(X, m) < g(m)$ pour tout $m$ et pour tout $X \in \mathcal{Q}_\sigma$. Comme $\mathcal{P}$ ne contient aucun point calculable, alors pour tout $\sigma$, $\mathcal{Q}_\sigma \subseteq \mathcal{P}$ non plus. Donc, d'après le corollaire 3.7, il doit exister $\tau_0, \tau_1$ incomparables telles que les classes $\mathcal{Q}_\sigma \cap [\tau_0]$ et $\mathcal{Q}_\sigma \cap [\tau_1]$ sont toutes les deux non vides. On définit alors $\mathcal{P}_{\sigma 0} = \mathcal{Q}_\sigma \cap [\tau_0]$ et $\mathcal{P}_{\sigma 1} = \mathcal{Q}_\sigma \cap [\tau_1]$.

Pour chaque $X \in 2^{\mathbb{N}}$, la classe $\bigcap_n \mathcal{P}_{X\restriction_n} \subseteq \mathcal{P}$ contient exactement un élément $G_X$ ; cet élément est calculatoirement dominé, et par construction $X \neq Y$ implique $G_X \neq G_Y$. La classe des $G_X$ pour $X \in 2^{\mathbb{N}}$ forme en fait un arbre parfait, dont les nœuds sont déterminés par le choix des extensions incomparables $\tau_0, \tau_1$. ∎

La technique de duplication du théorème précédent peut également être appliquée au théorème de base d'évitement de cône, mais il ne peut bien entendu pas être utilisé avec le théorème de la base low, car la classe des ensembles low est dénombrable. Le lecteur pourra essayer de l'appliquer quand même, afin de voir ce qui coince.

Notons enfin qu'il n'est bien entendu pas nécessaire de passer par des classes $\Pi^0_1$ pour construire une classe parfaite d'ensembles calculatoirement dominés, et l'on peut appliquer, comme suit, la même idée de duplication de construction à la preuve des f-arbres.

**Exercice 5.2.** (⋆) Construire une classe parfaite d'ensembles calculatoirement dominés via des f-arbres. ◇

**Exercice 5.3.** (⋆) Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide sans point calculable. Mélanger la construction ci-dessus avec la preuve du lemme 4.6 pour obtenir une sous classe parfaite de $\mathcal{P}$ dont tous les éléments sont calculatoirement dominés, et dont les degrés Turing sont deux à deux incomparables. ◇

**Exercice 5.4.** (⋆⋆) Soit $\mathcal{P}$ une classe parfaite. Construire une sous-classe parfaite de $\mathcal{P}$ dont les éléments sont deux à deux incomparables en termes de degrés Turing. ◇

**Exercice 5.5.** $(\star)$ Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide sans point isolé. Montrer que $\emptyset'$ calcule un élément non calculable de $\mathcal{P}$. $\diamond$

Notons que le dernier exercice utilise nécessairement le fait que la classe $\mathcal{P}$ ne contient pas de point isolé. Nous verrons avec la proposition 30-3.5 une technique simple, mais très puissante, permettant de construire des classes $\Pi^0_1$ —avec points isolés— dont les éléments sont soit des ensembles finis, soit des ensembles de complexité calculatoire « très élevée ».

## 6. Degrés PA

Nous prenons ici un peu d'avance sur le chapitre 9, dans lequel nous exposons les notions de théorie logique du premier ordre, de système formel de l'arithmétique de Peano ainsi que du premier théorème d'incomplétude de Gödel : la notion de degré PA est née en lien direct avec ces notions. Nous allons toutefois rapidement nous abstraire de cet aspect historique pour donner avec le théorème 6.2 une caractérisation équivalente des degrés PA ne faisant appel qu'aux notions de calculabilité vues jusqu'ici.

L'étude des degrés PA —acronyme de « Peano Arithmetic »— remonte aux travaux de Gödel et de son fameux théorème d'incomplétude : il n'existe pas d'extension calculable, complète et cohérente des axiomes de l'arithmétique de Peano [1]. L'étude des degrés Turing se développant, la question de la puissance nécessaire pour calculer une telle extension s'est tout naturellement posée. Nous allons voir que les développements autour de cette question ont abouti à l'un des concepts les plus riches de la calculabilité, qui a sans doute trouvé l'apogée de sa force et de son intérêt à travers l'étude des mathématiques à rebours.

Dans ce qui suit, fixons une énumération calculable $\psi_0, \psi_1, \psi_2, \dots$ de toutes les formules de l'arithmétique. Supposons également qu'il existe une fonction calculable $\texttt{neg} : \mathbb{N} \to \mathbb{N}$ telle que $\psi_{\texttt{neg}(n)} = \neg\psi_n$. Pour le théorème qui suit, nous appellerons *théorie* un ensemble $T \subseteq \mathbb{N}$ tel que pour tout $m$, si $\{\psi_n : n \in T\} \vdash \psi_m$, alors $m \in T$.

En d'autres termes, une théorie est un ensemble de formules de l'arithmétique clos par conséquence logique. Une théorie $T$ est *cohérente* si le code de la formule « $0 = 1$ » n'appartient pas à $T$. Une théorie $T$ est *complète* si pour tout $n$ : soit $n \in T$, soit $\texttt{neg}(n) \in T$. Le lecteur qui aborde ces notions pour la première fois trouvera plus de détails dans le chapitre 9.

1. Cette version du théorème est en fait un renforcement de celui de Gödel, qui fut établi par Rosser.

**Définition 6.1.** Une *complétion de l'arithmétique de Peano* est une théorie complète $T$ contenant $\{n \in \mathbb{N} \ : \ PA \vdash \psi_n\}$. Un degré Turing est *PA* s'il contient une complétion cohérente de l'arithmétique de Peano. ♢

Les degrés PA étant clos par le haut, il est équivalent pour un degré d'être PA et de contenir un ensemble qui calcule une complétion cohérente de l'arithmétique de Peano.

Nous montrons dès à présent une équivalence qui nous servira de caractérisation pour les degrés PA.

**Théorème 6.2 (Jockusch et Soare [96], Solovay (non publié))**
*Soit $X$ un ensemble. Les deux énoncés suivants sont équivalents.*

*(1) $X$ est de degré* PA.

*(2) $X$ est de degré* $\mathrm{DNC}_2$*, c'est-à-dire que l'ensemble $X$ calcule une fonction $f : \mathbb{N} \to \{0, 1\}$ telle que $f(n) \neq \Phi_n(n)$ pour tout $n$.*

Avant de passer à la preuve, nous renvoyons le lecteur à la définition 7-2.7, qui introduisait la notion de degré $\mathrm{DNC}_f$ pour une fonction $f : \mathbb{N} \to \mathbb{N}$ telle que $2 \leqslant f(n) \leqslant f(n+1)$. La notion de degré $\mathrm{DNC}_2$ est la plus forte possible de cet ordre : la fonction $f$ calculée ne dispose que de deux possibilités (0 ou 1) pour différer de chaque $\Phi_n(n)$. Nous verrons avec les corollaires 18-4.3 et 19-1.8 que de nombreux ensembles de degré DNC ne sont pas $\mathrm{DNC}_2$.

Preuve. L'équivalence montrée par Jockusch et Soare utilise le théorème de base de Scott [194] pour les degrés PA, qui stipule que tout degré PA calcule un chemin infini dans toute classe $\Pi^0_1$ non vide. Nous montrons ici directement l'équivalence.

L'implication (1) $\Rightarrow$ (2) est essentiellement le théorème de Gödel-Rosser, qui étend le premier théorème d'incomplétude de Gödel, et qui sera démontrée formellement avec le théorème 9-3.10 et le corollaire 9-3.11.

Montrons (2) $\Rightarrow$ (1). Soit $f \leqslant_T X$ une fonction à valeurs dans $\{0, 1\}$ telle que $f(n) \neq \Phi_n(n)$ pour tout $n$. On définit $T_0 = \mathrm{PA}$. Supposons $T_n$ cohérente, définie à l'étape $n$. On considère la formule de l'arithmétique $\psi_n$ de code $n$, et l'on définit le code de machine $e_n$ tel que

$$\Phi_{e_n}(e_n) = 1, \text{ si } T_n + \psi_n \vdash 0 = 1, \quad \text{et} \quad \Phi_{e_n}(e_n) = 0, \text{ si } T_n + \neg\psi_n \vdash 0 = 1.$$

Si $\Phi_{e_n}(e_n) \downarrow = 0$, alors $T_n + \neg\psi_n$ est incohérente, et $T_n + \psi_n$ est donc cohérente. Si $\Phi_{e_n}(e_n) \downarrow = 1$, alors $T_n + \psi_n$ est incohérente, et $T_n + \neg\psi_n$ est donc cohérente. Si $\Phi_{e_n}(e_n) \uparrow$, alors $T_n + \psi_n$ et $T_n + \neg\psi_n$ sont toutes les deux cohérentes. À présent, comme $f(e_n) \neq \Phi_{e_n}(e_n)$, on peut ainsi définir $T_{n+1} = T_n + \psi_n$ si $f(e_n) = 1$ et $T_{n+1} = T_n + \neg\psi_n$ si $f(e_n) = 0$. Dans tous les cas, on aura une théorie cohérente.

La théorie $T = \bigcup_n T_n$ est donc cohérente, et elle est par construction également complète. ∎

**Remarque**

Notons que la direction (2) → (1) du théorème 6.2 fonctionne pour toute théorie $T_0$ cohérente dont les axiomes sont calculables. Ainsi, toute fonction $\mathrm{DNC}_2$ est capable de calculer une complétion de toute théorie cohérente dont les axiomes sont calculables. La direction (1) → (2) est plus spécifique à l'arithmétique de Peano, car elle requiert une théorie suffisamment expressive pour coder des calculs par des formules.

Notons que $\emptyset'$ peut calculer une fonction $\mathrm{DNC}_2$ et est donc de degré PA. La proposition suivante permet de déduire qu'il n'est pas du tout nécessaire d'être Turing complet pour calculer une extension complète et cohérente de l'arithmétique de Peano.

**Définition 6.3.** Le *spectre de degrés* d'une classe $\mathcal{P} \subseteq 2^{\mathbb{N}}$ est l'ensemble

$$\deg \mathcal{P} = \{\deg_T X : X \in [P]\}.$$ ♢

**Proposition 6.4.** Il existe une classe $\Pi^0_1$ dont le spectre de degrés correspond aux degrés PA. ⋆

Preuve. Il s'agit d'une simple constatation, qui a déjà été utilisée pour la preuve du théorème 3.5. La classe des ensembles $\mathrm{DNC}_2$ se décrit de la manière suivante :

$$\mathcal{P} = \{X \in 2^{\mathbb{N}} \ : \ \forall e \ \forall t \ \Phi_e(e)[t] \uparrow \vee \ \Phi_e(e)[t] \downarrow \neq X(e)\}.$$ ∎

**Corollaire 6.5**
*Il existe des degrés PA qui sont low.*

Preuve. D'après la proposition 6.4 et le théorème 4.3. ∎

**Corollaire 6.6**
*Il existe des degrés PA calculatoirement dominés.*

Preuve. D'après la proposition 6.4 et le théorème 4.5. ∎

**Corollaire 6.7**
*Soit $A$ un ensemble non calculable. Alors, il existe un ensemble $X$ de degré PA qui ne calcule pas $A$.*

Preuve. D'après la proposition 6.4 et le théorème 4.7. ∎

Voyons à présent une autre caractérisation importante des degrés PA, qui stipule que ces derniers capturent la puissance de calcul nécessaire et suffisante pour le lemme faible de König.

**Théorème 6.8**
*Soit $X \subseteq \mathbb{N}$. Les deux énoncés suivants sont équivalents.*

*(1) $X$ est de degré PA.*

*(2) $X$ calcule un ensemble dans chaque classe $\Pi^0_1$ non vide.*

*De plus, pour (2), le calcul est uniforme en un code de la classe $\Pi^0_1$.*

Preuve. Pour (2) $\Rightarrow$ (1), il suffit de remarquer qu'il existe une classe $\Pi^0_1$ non vide ne contenant que des ensembles de degrés PA (voir la proposition 6.4).

Montrons à présent (1) $\Rightarrow$ (2). Soit $f \leqslant_T X$ à valeurs dans $\{0,1\}$, telle que $f(n) \neq \Phi_n(n)$ pour tout $n$. Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide, et soit $T$ un arbre calculable tel que $[T] = \mathcal{P}$. Soit $\sigma_0 = \epsilon$. Étant donné $\sigma_n$ défini tel que $[\sigma_n] \cap [T]$ est non vide, on calcule $\sigma_{n+1} = \sigma_n i$ pour $i \in \{0,1\}$ de la manière suivante : on calcule d'abord le code $e_n$ d'un programme qui sur toute entrée $m$ cherche le plus petit $t$ tel que pour $i = 0$ ou $i = 1$ aucune chaîne $\sigma$ de taille $t$ avec $\sigma \succeq \sigma_n i$ n'appartient à $T$. D'après le lemme de König, cette condition est équivalente au fait que $[\sigma_n i] \cap [T]$ soit vide. Si le programme repère un des deux événements, il s'arrête avec comme valeur $i$. Il suffit alors de regarder la valeur de $f(n)$. On définit simplement $\sigma_{n+1} = \sigma_n f(n)$. Comme $f(e_n) \neq \Phi_{e_n}(e_n)$, on a la garantie que $[\sigma_{n+1}] \cap [T]$ est non vide. ∎

**Classe universelle**

Notons que d'après la proposition 6.4, il existe une classe $\Pi^0_1$ non vide dont tous les membres sont de degré PA, et que d'après le théorème 6.8, tout degré PA calcule un membre de chaque classe $\Pi^0_1$ non vide. Une telle classe est donc « maximale » en termes de complexité calculatoire, au sens où si l'on sait calculer un membre de cette classe, alors on sait calculer un membre de toute classe $\Pi^0_1$ non vide.
On appelle *classe* $\Pi^0_1$ *universelle* une classe $\Pi^0_1$ non vide dont tous les membres sont de degré PA.

Dans la même veine que le théorème 7-7.1, on termine par une caractérisation qui combine à présent le fait d'être de degré high ou PA. Notez la différence avec (1) $\leftrightarrow$ (3) du théorème 7-6.2, au sein duquel on considère une suite $(X_n)_{n\in\mathbb{N}}$ contenant exactement les ensembles calculables, alors qu'ici l'on considère seulement qu'elle contient les ensembles calculables.

**Théorème 6.9 (Jockusch [98])**
*Soit $X \subseteq \mathbb{N}$. Les deux énoncés suivants sont équivalents.*

*(1) L'ensemble $X$ est de degré high ou PA.*

*(2) L'ensemble $X$ calcule une suite $(X_n)_{n\in\mathbb{N}}$ contenant tous les ensembles calculables.*

Preuve. Montrons d'abord (1) implique (2). Si $X$ est high, alors l'implication est claire d'après le théorème 7-6.2. Supposons à présent que $X$ est de degré PA. Soit $g \leqslant_T X$ telle que $g(n) \neq \Phi_n(n)$ pour tout $n$. Notons que l'ensemble $X$ calcule aussi la fonction $f$ qui inverse les valeurs de $g$, c'est-à-dire telle que $\Phi_n(n)\downarrow \in \{0,1\}$ implique $f(n) = \Phi_n(n)$. Étant donné une fonction calculable $\Phi_e$ et un entier $n$, on peut calculer le code $a_n$ tel que $\Phi_{a_n}(a_n) = \Phi_e(n)$. En utilisant ce procédé et le fait que $\Phi_{a_n}(a_n)\downarrow \in \{0,1\}$ implique $f(a_n) = \Phi_{a_n}(a_n) = \Phi_e(n)$, on calcule aisément un ensemble $X_e$ tel que si $n \mapsto \Phi_e(n)$ est totale et à valeurs dans $\{0,1\}$ alors $X_e(n) = \Phi_e(n)$ pour tout $n$. On peut donc calculer notre suite $(X_e)_{e\in\mathbb{N}}$ contenant tous les ensembles calculables.

Montrons à présent (2) implique (1). Soit $(X_n)_{n\in\mathbb{N}}$ une suite $X$-calculable contenant tous les ensembles calculables. L'idée est de procéder au départ comme dans la preuve de (3) $\Rightarrow$ (1) du théorème 7-6.2. Étant donné un prédicat $\Pi^0_2$ de la forme

$$P = \{e \ : \ \forall x_1 \ \exists x_2 \ R(e, x_1, x_2)\},$$

l'idée était de définir uniformément en $e$ une fonction partielle calculable $f_e$ telle que :

(a) $e \in P$ implique que $f_e$ est une fonction totale calculable ;

(b) $e \notin P$ implique que $f_e$ est une fonction partielle qui n'a aucune complétion totale calculable.

Il suffit de remarquer que dans le théorème 7-6.2, la définition de $f_e$ qui est donnée est telle que dans le cas (b), non seulement aucune complétion de $f_e$ n'est calculable, mais en plus une telle complétion est forcément de degré PA. La définition était la suivante.
Soit $e$ fixé. À l'étape de calcul $t$, pour toute valeur $n$ plus petite que $t$ et telle que $f_e$ ne n'arrête pour le moment pas sur $n$, on procède comme suit : si $\Phi_n(n)[t]\downarrow \neq 0$, on définit $f_e(n) = 0$. Sinon, si $\Phi_n(n)[t]\downarrow \neq 1$, on définit $f_e(n) = 1$. Sinon, si pour tous $k \leqslant n$ il existe $m_k \leqslant t$ tel que $R(e, k, m_k)$, alors on définit $f_e(n) = 0$.

Tout comme dans la preuve du théorème 7-6.2, si $e \in P$ alors $f_e$ est une fonction totale. Dans le cas contraire, on s'aperçoit que pour presque toutes les valeurs de $n$ telles que $\Phi_n(n)\downarrow$ on a $f_e(n) \neq \Phi_n(n)$. Toute complétion

de $f_e$ est donc une fonction $\mathrm{DNC}_2$, modulo un nombre fini de valeurs, et est donc de degré PA.

Il y a à présent deux possibilités :

▷ soit $(X_n)_{n\in\mathbb{N}}$ contient un ensemble de degré PA, auquel cas on a (1) ;

▷ soit ce n'est pas le cas, et l'on peut alors donner une définition $\Sigma^0_2(X)$ de $P$ comme dans la preuve du théorème 7-6.2, ce qui implique, appliquée à $P = \mathbb{N} \setminus \emptyset''$, que $\emptyset''$ est $\Delta^0_2(X)$, et que $X$ est donc high. ∎

Nous terminons cette section par un exercice qui constitue une caractérisation alternative et bien connue des degrés PA.

**Exercice 6.10. (⋆)** Montrer que $X$ est PA si, et seulement si, pour tous ensembles c. e. $A, B$ avec $A \cap B = \emptyset$, il existe un ensemble $X$-calculable $C$ tel que $A \subseteq C$ et $C \cap B = \emptyset$. ◇

# 7. Arbres à branchement fini

Nous introduisons ici l'*espace de Baire* : la classe $\mathbb{N}^{\mathbb{N}}$ de toutes les suites infinies à valeurs dans $\mathbb{N}$, ou autrement dit la classe de toutes les fonctions de $\mathbb{N}$ dans $\mathbb{N}$. Tout comme l'espace de Cantor a son ensemble de chaînes $2^{<\mathbb{N}}$, l'espace de Baire a son ensemble de chaînes $\mathbb{N}^{<\mathbb{N}}$ : les suites finies à valeurs dans $\mathbb{N}$. Les différentes opérations que nous avons vues sur les chaînes binaires (préfixe, concaténation, longueur, ...) s'étendent sans problème aux chaînes de l'espace de Baire. En particulier, étant donné une chaîne $\sigma \in \mathbb{N}^{<\mathbb{N}}$, on dénote par $[\sigma]$ la classe des suites $P \in \mathbb{N}^{\mathbb{N}}$ telles que $\sigma \prec P$. La notion d'arbre s'étend également aux sous-ensembles de $\mathbb{N}^{<\mathbb{N}}$ comme suit.

**Définition 7.1.** Un ensemble $T \subseteq \mathbb{N}^{<\mathbb{N}}$ est un *arbre* si $T$ est clos par préfixe, c'est-à-dire pour tout $\sigma \in T$ et tout $\tau \preceq \sigma$, alors $\tau \in T$. ♢

Contrairement aux arbres binaires, les nœuds peuvent avoir une infinité de successeurs. Un nœud $\sigma \in T$ est *branchant* s'il a au moins deux successeurs. Un *chemin* de $T$ est une suite $P \in \mathbb{N}^{\mathbb{N}}$ dont tous les segments initiaux sont dans $T$. On dénote par $[T]$ la classe des chemins de $T$. Le lemme de König ne fonctionne plus sur les arbres de l'espace de Baire, comme le montre contre-exemple suivant.

**Exemple 7.2.** Soit $T = \{\sigma \in \mathbb{N}^{<\mathbb{N}} : \forall n < |\sigma|\ \sigma(n) \geqslant |\sigma|\}$. L'arbre $T$ contient des nœuds de longueur arbitraire et est infini, mais $[T] = \emptyset$.

La puissance calculatoire des chemins d'un arbre calculable arbitraire de l'espace de Baire sera étudiée dans la partie IV sur l'hypercalculabilité. Dans cette section, nous allons nous restreindre à une sous-catégorie de ces arbres tombant sous la coupe du lemme de König. Les arbres $T \subseteq \mathbb{N}^{<\mathbb{N}}$ à *branchement fini*, c'est-à-dire au sein desquels chaque nœud a un nombre fini de successeurs.

**Lemme 7.3 (Lemme de König).** Soit $T \subseteq \mathbb{N}^{<\mathbb{N}}$ un arbre à branchement fini tel que $|T| = \infty$. Alors, $[T]$ est non vide. ⋆

Preuve. On construit un chemin $X$ par récurrence sur $n$. Comme $T$ est infini, mais que la racine $\epsilon$ n'a qu'un nombre fini de successeurs, il existe d'après le principe des tiroirs un $i \in \mathbb{N}$ et une infinité de nœuds $\sigma \in T$ qui étendent $i$ (c'est-à-dire avec $i \prec \sigma$). On définit $X(0) = i$. Supposons que $\tau = X(0)X(1)\dots X(n)$ soit défini avec $\tau \in T$ et tel qu'il y a une infinité de nœuds $\sigma \in T$ pour lesquels $\tau \preceq \sigma$. Le nœud $\sigma$ n'ayant qu'un nombre fini de successeurs, le principe des tiroirs confirme qu'il existe $i \in \mathbb{N}$ et une infinité de nœuds $\sigma \in T$ qui étendent $\tau i$. On définit $X(n+1) = i$.

Par récurrence sur $n$, on définit donc de cette manière un ensemble $X$ tel que $X\restriction_n \in T$ pour tout $n$. ∎

**L'espace de Baire**

Comme pour l'espace de Cantor, les ouverts de l'espace de Baire sont les classes $\mathcal{O} \subseteq \mathbb{N}^{\mathbb{N}}$ de la forme $\mathcal{O} = \bigcup_{\sigma \in W} [\sigma]$ pour un ensemble $W \subseteq \mathbb{N}^{<\mathbb{N}}$, et les fermés $\mathcal{P} \subseteq \mathbb{N}^{\mathbb{N}}$ sont de la forme $[T]$ pour un arbre $T \subseteq \mathbb{N}^{<\mathbb{N}}$. En revanche, contrairement à l'espace de Cantor, les fermés de l'espace de Baire ne sont pas compacts en général. Les compacts de l'espace de Baire sont précisément les fermés $\mathcal{P}$ de la forme $[T]$ pour un arbre $T \subseteq \mathbb{N}^{<\mathbb{N}}$ à branchement fini.

La preuve du lemme de König est quasiment la même que celle de sa version faible, et l'on pourrait s'attendre à première vue à ce que la puissance calculatoire nécessaire pour calculer un chemin d'un arbre calculable à branchement fini soit celle des degrés PA. Ce n'est cependant pas le cas, comme le montre la proposition 7.4.

**Proposition 7.4.** Soit $T \subseteq 2^{<\mathbb{N}}$ un arbre binaire $\Delta^0_2$. Il existe un arbre calculable à branchement fini $S$ tel que $\deg([T]) = \deg([S])$. ⋆

Preuve. Soit $(T_n)_{n\in\mathbb{N}}$ une approximation $\Delta^0_2$ de l'arbre $T$. On peut supposer sans perte de généralité que pour tout entier $n$, $T_n$ est clos par préfixe et $T_n \subseteq 2^{\leqslant n}$ (l'ensemble des chaînes de taille inférieure ou égale à $n$). On montre aisément que toute réunion d'arbres est un arbre, ce qui implique que $\bigcup_n T_n$ est un arbre.

Montrons que $[\bigcup_n T_n] = [T]$. Clairement, $T \subseteq \bigcup_n T_n$, donc $[T] \subseteq [\bigcup_n T_n]$. Soit $P \in [\bigcup_n T_n]$. Soit $s$ un entier $\geqslant 0$ ; montrons que $P\restriction_s \in T$. L'approximation $(T_n)_{n\in\mathbb{N}}$ de $T$ étant $\Delta^0_2$, on a $P\restriction_s \in T$ ssi $\forall t\ \exists n \geqslant t\ P\restriction_s \in T_n$. Soit $t \geqslant s$. Comme $P\restriction_t \in \bigcup_n T_n$ et comme par hypothèse $\bigcup_{n<t} T_n \subseteq 2^{<t}$, on a $P\restriction_t \in T_n$ pour $n \geqslant t$. Par clôture par le bas de $T_n$, il vient $P\restriction_s \in T_n$. Donc, $\forall t\ \exists n \geqslant t\ P\restriction_s \in T_n$. Ainsi, $P\restriction_s \in T$.

Soient $\sigma, \tau \in \mathbb{N}^{<\mathbb{N}}$ de même longueur. On note $\langle \sigma, \tau\rangle$ la chaîne $\rho$ de longueur $|\sigma|$ telle que pour tout $n < |\rho|$, $\rho(n) = \langle \sigma(n), \tau(n)\rangle$. L'opération s'étend naturellement aux suites infinies $P, Q$ pour lesquelles on écrira $\langle P, Q\rangle$.

Nous allons construire un arbre calculable $S \subseteq \mathbb{N}^{<\mathbb{N}}$ à branchement fini dont tous les chemins seront de la forme $\langle P, Q\rangle$ avec $P \in [\bigcup_n T_n] = [T]$ et $Q$ un « témoin » de $P \in [\bigcup_n T_n]$, au sens où pour tout $s$, $P\restriction_s \in T_{Q(s)}$.

Définissons une fonction partielle calculable $f : \bigcup_n T_n \to \mathbb{N}^{<\mathbb{N}}$ qui envoie des chaînes vers des chaînes de même longueur inductivement comme suit.

▷ $f(\epsilon) = \epsilon$.

▷ Si $\sigma i \in \bigcup_n T_n$, alors $f(\sigma i) = f(\sigma)^\frown s$, où $s$ est le plus petit entier tel que $\sigma i \in T_s$ et $^\frown$ la concaténation.

▷ Par continuité, la fonction $f$ s'étend aux suites infinies de $[\bigcup_n T_n] = [T]$.

Soit $S = \{\langle \sigma, f(\sigma)\rangle : \sigma \in \bigcup_n T_n\}$. Notons que la réunion $\bigcup_n T_n$ n'est pas calculable en général, mais que $S$ l'est, car il est facile de vérifier que pour tout $\rho = \langle \sigma, \mu\rangle$, on a $f(\sigma) = \mu$. L'ensemble $S$ est clos par préfixe, car $\bigcup_n T_n$ l'est également et $f(\sigma i)\restriction_{|\sigma|} = f(\sigma)$ pour tout $\sigma \in \bigcup_n T_n$. Ainsi, $S$ est un arbre calculable. Notons également que $S$ est 2-branchant, donc à branchement fini.

Montrons que $\deg([T]) = \deg([S])$. Soit $P \in [T]$ ; alors, $\langle P, f(P)\rangle \in [S]$ et $P \equiv_T \langle P, f(P)\rangle$. Soit $R \in [S]$ ; alors, $R = P \oplus f(P)$ pour un $P$ dans $[\bigcup_n T_n] = [T]$. De même, $R \equiv_T P$. Cela conclut la preuve de la proposition 7.4. ∎

**Corollaire 7.5**
*Il existe un arbre calculable $T \subseteq \mathbb{N}^{<\mathbb{N}}$ à branchement fini et un degré PA $P$ ne calculant pas de chemin à travers $T$.*

Preuve. Soit $S = \{\emptyset'\restriction_n : n \in \mathbb{N}\}$ l'arbre binaire $\Delta^0_2$ ayant $\emptyset'$ pour unique chemin infini. Par la proposition 7.4, il existe un arbre calculable $T \subseteq \mathbb{N}^{<\mathbb{N}}$ à branchement fini tel que $\deg([T]) = \deg([S])$. En particulier, tout chemin de $T$ calcule $\emptyset'$. Par le corollaire 6.5, il existe un degré à la fois PA et low. En particulier, ce degré ne calcule pas de chemin à travers $T$. ∎

On peut relativiser la notion d'être $\mathrm{DNC}_2$ relativement à un oracle $X$ : on demande le calcul d'une fonction $f : \mathbb{N} \to \{0,1\}$ telle que $f(n) \neq \Phi_n(X,n)$ pour tout $n$. Le théorème 6.8 se relativise bien au sens où les degrés $\mathrm{DNC}_2$ relativement à $X$, que l'on appellera aussi degrés PA relativement à $X$ ou $\mathrm{PA}(X)$, coïncident avec ceux permettant de calculer un chemin dans toute classe $\Pi^0_1(X)$ non vide.

**Exercice 7.6.** Soit $Y$ un ensemble $\mathrm{PA}(X)$. Montrer que $Y \geqslant_T X$. ◇

On en déduit qu'un degré PA relativement à $\emptyset'$ est nécessaire pour calculer un chemin dans tout arbre calculable infini à branchement fini. Notons que la situation de l'exercice 7.6 est différente lorsque l'on considère les degrés DNC au lieu des degrés $\mathrm{DNC}_2$. Plus précisément, si $X$ est un ensemble non calculable, il existe un ensemble $Y$ de degré DNC relativement à $X$ qui ne calcule pas $X$ (voir le corollaire 18-4.4). Ce résultat fait intervenir des notions de théorie de l'aléatoire que nous aborderons au chapitre le chapitre 18.

**Arbre binaire vs arbre 2-branchant**

L'arbre $T \subseteq \mathbb{N}^{<\mathbb{N}}$ construit dans la preuve de la proposition 7.4 est 2-branchant, au sens où chaque nœud a au plus deux successeurs. D'un point de vue purement structurel, il est donc isomorphe à un arbre binaire $S \subseteq 2^{<\mathbb{N}}$. Pourtant, il existe des degrés PA qui ne calculent pas de chemin dans cet arbre. La différence entre la puissance calculatoire de cet arbre et celle d'un arbre binaire ne provient donc pas d'une différence combinatoire, mais relève tout simplement d'un manque d'information sur les successeurs d'un nœud : étant donné un arbre calculable à branchement fini, on ne peut pas borner de manière calculable et uniforme la valeur maximale du successeur d'un nœud.

La remarque précédente nous conduit à la définition qui suit.

**Définition 7.7.** Un arbre $T \subseteq \mathbb{N}^{<\mathbb{N}}$ est *calculatoirement borné* s'il existe une fonction calculable $g : \mathbb{N} \to \mathbb{N}$ telle que pour tout $\sigma \in T$, et pour tout $n < |\sigma|$, on a l'inégalité $\sigma(n) < f(n)$. ♢

Il est clair que tout arbre calculatoirement borné est à branchement fini. La proposition suivante permet de réconcilier l'idée selon laquelle des objets combinatoirement similaires devraient avoir la même puissance calculatoire, en montrant que dès que l'arbre à branchement fini est accompagné d'une borne calculable sur son branchement, alors la puissance calculatoire nécessaire pour calculer un chemin est exactement celle des degrés PA.

Étant donné une fonction $f : \mathbb{N} \to \mathbb{N}$, nous noterons $f^{<\mathbb{N}}$ l'ensemble des chaînes $\sigma \in \mathbb{N}^{<\mathbb{N}}$ telles que pour tout $n < |\sigma|$, $\sigma(n) < f(n)$.

**Proposition 7.8.** Pour tout arbre $T \subseteq \mathbb{N}^{<\mathbb{N}}$ calculable et calculatoirement borné, il existe un arbre binaire $S \subseteq 2^{<\mathbb{N}}$ tel que $\deg([T]) = \deg([S])$. ⋆

Preuve. L'idée de la preuve est tout simplement de définir un codage binaire des chaînes, en utilisant la borne calculatoire pour savoir combien de bits allouer à chaque niveau. Pour retirer toute ambiguïté, nous noterons $2^{=n}$ l'ensemble des chaînes binaires de longueur $n$, au lieu de $2^n$, qui désignera la $n$-ième puissance de 2. Soit $g : \mathbb{N} \to \mathbb{N}$ une fonction calculable telle que $T \subseteq g^{<\mathbb{N}}$. Sans perte de généralité, nous pouvons supposer que $g(n) = 2^{h(n)}$ pour tout $n$, avec $h : \mathbb{N} \to \mathbb{N}$ une fonction calculable.

Pour tout entier $n$, soit $e_n : 2^n \to 2^{=n}$ la bijection canonique. Par exemple, la fontion $e_2 : 4 \to \{00, 01, 10, 11\}$ vérifie

$$e_2(0) = 00, \quad e_2(1) = 01, \quad e_2(2) = 10 \quad \text{et} \quad e_3(3) = 11.$$

Ce codage s'étend en une bijection calculable $e : g^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ définie par

$$e(\sigma) = e_{h(0)}(\sigma(0))^\frown e_{h(1)}(\sigma(1))^\frown \cdots {}^\frown e_{h(|\sigma|-1)}(\sigma(|\sigma| - 1)),$$

où l'on dénote ici —comme d'usage— la concaténation par $^\frown$. Par exemple, si $h(n) = n + 1$, alors $g(n) = 2^{h(n)} = 2^{n+1}$, et

$$e(032) = e_1(0)^\frown e_2(3)^\frown e_3(2) = 0^\frown 11^\frown 010 = 011010.$$

Notons que l'ensemble $\widehat{S} = \{e(\sigma) : \sigma \in T\}$ n'est pas un arbre, car il n'est clos par préfixe que pour les segments initiaux de longueur exactement $\operatorname{Im} g$. Nous devons donc définir l'arbre $S$ comme la clôture par préfixe de $\widehat{S}$, autrement dit $S = \{e(\sigma)\restriction_n : \sigma \in T \wedge n \in \mathbb{N}\}$. La fonction de codage $e$ étant monotone sur les longueurs, et l'ensemble $T$ étant clos par préfixe, $\rho \in S$ si, et seulement si, il existe une chaîne $\sigma \in g^{<\mathbb{N}}$ de longueur au plus $|\rho|$ telle que $\rho \prec e(\sigma)$. Ainsi, $S$ est un arbre binaire calculable infini, dont les chemins sont exactement les suites infinies de la forme $e_{h(0)}(P(0))^\frown e_{h(1)}(P(1))^\frown \ldots$, pour $P \in [T]$. Dès lors, $\deg([T]) = \deg([S])$. ∎

**Corollaire 7.9**
*Soit $T \subseteq \mathbb{N}^{<\mathbb{N}}$ un arbre infini calculable et calculatoirement borné. Tout degré PA calcule un chemin de $T$.*

Preuve. Par la proposition 7.8, il existe un arbre binaire $S \subseteq 2^{<\mathbb{N}}$ tel que $\deg([T]) = \deg([S])$. Par le théorème 6.8, tout degré PA calcule un chemin de $S$, donc tout degré PA calcule un chemin de $T$. ∎

Nous voyons à présent la réciproque de la proposition 7.4, qui montre que les degrés PA relativement à $\emptyset'$ sont exactement ceux permettant de calculer un chemin dans un arbre calculable à branchement fini.

**Proposition 7.10.** Soit $T \subseteq \mathbb{N}^{<\mathbb{N}}$ un arbre calculable infini à branchement fini. Il existe $S \subseteq 2^{<\mathbb{N}}$ un arbre binaire $\Delta^0_2$ tel que $\deg([T]) = \deg([S])$. ⋆

Preuve. Notons tout d'abord que tout arbre calculable infini à branchement fini $T \subseteq \mathbb{N}^{<\mathbb{N}}$ est $\emptyset'$-calculatoirement borné, c'est-à-dire qu'il existe une fonction $\emptyset'$-calculable $g : \mathbb{N} \to \mathbb{N}$ telle que $T \subseteq g^{<\mathbb{N}}$. La preuve de la proposition 7.8 se relativise à $\emptyset'$, et permet de définir $S \subseteq 2^{<\mathbb{N}}$ un arbre binaire $\Delta^0_2$ tel que $\deg([T]) = \deg([S])$. ∎

Donnons pour finir quelques exercices. La *clôture par préfixe* d'un ensemble $S \subseteq \mathbb{N}^{<\mathbb{N}}$ est l'ensemble

$$\widehat{S} = \{\tau \in \mathbb{N}^{<\mathbb{N}} \ : \ \exists \sigma \in S \ \tau \preceq \sigma\}.$$

**Exercice 7.11.** Montrer que pour tout ensemble infini $S \subseteq 2^{<\mathbb{N}}$, sa clôture par préfixe admet un chemin. ◇

**Exercice 7.12.** Montrer qu'il existe un ensemble infini calculable $S \subseteq 2^{<\mathbb{N}}$ tel que $\left[\widehat{S}\right] = \{\emptyset'\}$. ◇

**Exercice 7.13.** **(⋆⋆)** Montrer que pour tout arbre $T \subseteq 2^{<\mathbb{N}}$ infini et $\emptyset'$-calculable, il existe un ensemble infini $S \subseteq 2^{<\mathbb{N}}$ contenant exactement une chaîne de chaque longueur, tel que $[T] = \left[\widehat{S}\right]$. ◇

## Schéma Récapitulatif

Voici une figure qui récapitule les différentes notions abordées jusqu'ici.

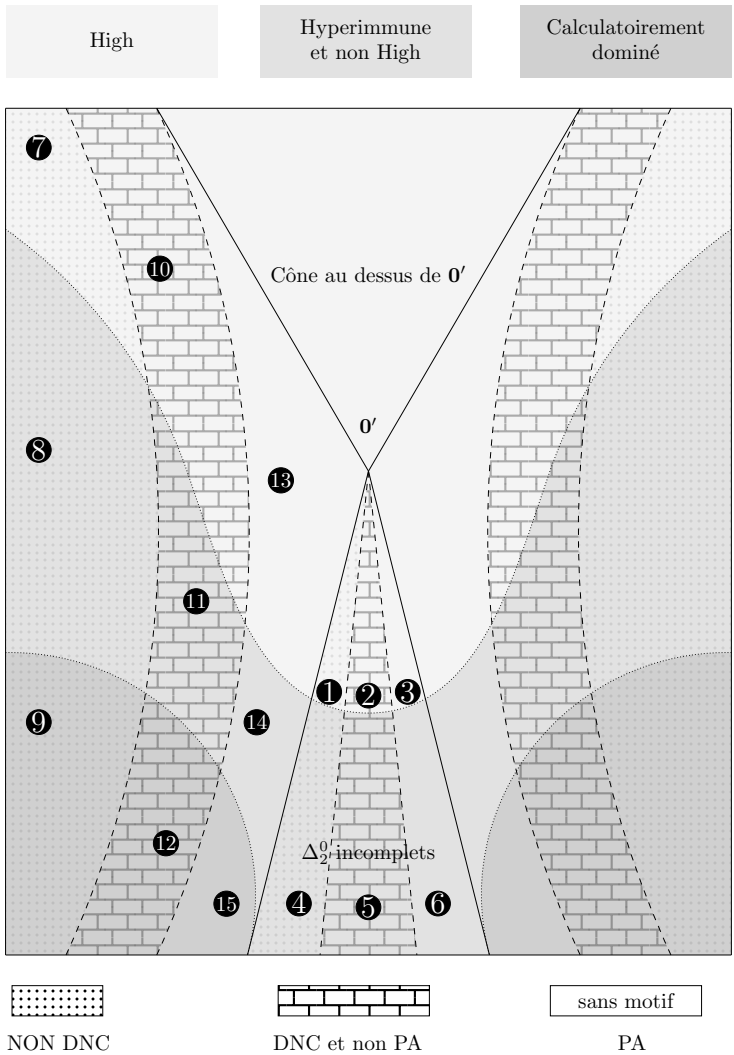



FIGURE 7.14 – *Récapitulatif sur les degrés Turing vus jusqu'ici. Les points (1) à (6) renvoient vers des exemples d'ensembles de chaque type. Les points de (7) à (15) renvoient vers l'existence d'une classe parfaite d'ensembles de chaque type (et donc d'après l'exercice 5.4 vers l'existence d'une classe parfaite de degrés Turing de chaque type).*

Les points de (1) à (15) qui suivent utilisent pour certains des concepts qui ne seront vus que dans les chapitres suivants.

(1) Il existe un degré Turing $\Delta^0_2$ non DNC et high : il suffit de mixer la preuve de l'exercice 4-10.6 avec celle de l'exercice 7-2.9 pour obtenir un ensemble high qui en plus d'être incomplet, ne soit pas DNC.

(2) Il existe un degré Turing $\Delta^0_2$ DNC, non PA et high : On part d'un ensemble $X$ DNC, non PA et low (par le théorème 18-4.1 et le corollaire 18-2.3, on peut prendre par exemple un aléatoire au sens de Martin-Löf low). On peut ensuite broder sur la construction de l'exercice 4-10.6 et sur celle aussi de l'exercice 7-2.9 pour construire un ensemble $Y$ $\Delta^0_2$ high tel que $X \oplus Y$ est non PA (en utilisant en particulier le fait que $X' \leqslant_T \emptyset'$).

(3) Il existe un degré Turing $\Delta^0_2$ incomplet, PA et high : on part d'un ensemble $X$ PA et low (voir le corollaire 6.5). On brode ensuite sur la construction de l'exercice 4-10.6 pour construire un ensemble $Y$ $\Delta^0_2$ high tel que $X \oplus Y$ est incomplet.

(4) Il existe un degré Turing $\Delta^0_2$ non DNC, hyperimmune et non high : il suffit de construire un ensemble non DNC et low, en mélangeant la proposition 4-9.1 et l'exercice 7-2.9.

(5) Il existe un degré Turing $\Delta^0_2$ DNC, non PA, hyperimmune et non high : il suffit de considérer un ensemble aléatoire au sens de Martin-Löf et low. D'après le théorème 18-4.1, un tel ensemble est DNC. D'après le théorème 19-1.7, il n'est pas PA. D'après la proposition 7-4.7, il est hyperimmune. Enfin, comme il est low il ne peut être high.

(6) Il existe un degré Turing $\Delta^0_2$ PA, hyperimmune et non high : d'après la proposition 7-4.7, il suffit de considérer un degré PA low donné par le corollaire 6.5.

(7) Il existe une classe parfaite d'ensembles high et non DNC. Il suffit d'utiliser le théorème 10-3.21 et de broder sur Posner/Robinson (voir le corollaire 10-3.34) pour construire un classe parfaite d'ensembles 1-génériques et high.

(8) Il existe une classe parfaite d'ensembles hyperimmunes, non high et non DNC. D'après le théorème 10-3.2, la proposition 10-3.38 et le corollaire 10-3.34 tout ensemble suffisamment générique sera dans ce cas-là.

(9) Il existe une classe parfaite d'ensembles calculatoirement dominés et non DNC. Il faut reprendre la construction d'ensembles calculatoirement dominés via des f-arbres, et la modifier pour produire des ensembles non DNC à la manière dont cela est fait dans l'exercice 7-2.9.

(10) Il existe une classe parfaite d'ensembles high, DNC et non PA. On peut appliquer le théorème 18-3.4 de Kučera/Gács relativisé à $\emptyset'$ sur l'arbre des 2-aléatoires pour construire des ensembles high et 2-aléatoires. Par le théorème 18-4.1, de tels ensembles sont DNC. Par le théorème 19-1.7, ils ne sont pas PA.

(11) Il existe une classe parfaite d'ensembles hyperimmunes, non high, DNC et non PA. D'après le théorème 18-4.1, le corollaire 19-3.9 et le corollaire 19-1.8, c'est le cas pour tout ensemble suffisamment aléatoire.

(12) Il existe une classe parfaite d'ensembles calculatoirement dominés, DNC et non PA. Il s'agit du théorème 5.1 appliqué à une classe $\Pi^0_1$ ne contenant que des aléatoires au sens de Martin-Löf. D'après le théorème 19-1.7, les ensembles MLR et calculatoirement dominés ne peuvent être PA.

(13) Il existe une classe parfaite d'ensembles incomplets, high et PA. On fixe un ensemble $X$ high et incomplet. On élabore ensuite sur le théorème 4.7 de base d'évitement de cône relativisé à $X$, pour construire une classe parfaite d'ensembles PA $Y$ tel que $X \oplus Y$ est incomplet.

(14) Il existe une classe parfaite d'ensembles hyperimmunes, non high et PA. Soit $X$ un ensemble hyperimmune et non high. On utilise la relativisation à $X$ du théorème 5.1, appliqué à la classe $\Pi^0_1$ des ensembles $\mathrm{DNC}_2$, pour construire une classe parfaite d'ensembles $Y$ tels que toute fonction calculée par $X \oplus Y$ est dominée par une fonction calculée par $X$.

(15) Il existe une classe parfaite d'ensembles calculatoirement dominés et PA. Il s'agit du théorème 5.1 appliqué à la classe $\Pi^0_1$ des ensembles $\mathrm{DNC}_2$.

Chapitre 9

# Interlude formel

## 1. Un peu d'histoire : la crise des fondements

Les mathématiques se sont développées naturellement au fil des siècles en tant qu'outil au service d'une représentation abstraite de la réalité. Cet état de fait est par exemple flagrant en sciences physiques pour lesquelles les mathématiques rendent compte avec précision d'un ensemble varié de phénomènes. Avec le temps, les notions étudiées sont devenues de plus en plus complexes, de plus en plus abstraites, et les connexions entre les mathématiques et le monde réel sont devenues de plus en plus incertaines. Pendant longtemps, la discipline a cependant pu compter sur le sens logique inné de l'esprit humain pour parler de choses que l'on ne « voyait plus » tout en gardant un cadre rigoureux. Les nombres complexes en constituent un exemple frappant : les nombres de carré négatif qui n'existent *a priori* que dans l'imagination du mathématicien sont baptisés en 1545 par Cardan « quantités sophistiquées ». De la sophistication il en fallait sans doute pour accepter cet ovni comme objet d'étude sérieux. Pourtant ces « quantités sophistiquées » trouvent leur utilité dans la résolution de problèmes, eux, très concrets. Raphaël Bombelli en donne une première formalisation en 1572, et montre comment utiliser ces nombres pour résoudre certaines équations du troisième degré. Ils trouveront au fil des siècles de nombreuses utilités en mathématiques, ainsi qu'en physique, où ils sont utilisés avec succès au sein d'équations représentant le monde réel.

Il a fallu un certain saut conceptuel pour accepter le développement d'un cadre mathématique rigoureux et cohérent autour des nombres complexes. Disons, malgré tout, que ce concept pour aussi surprenant qu'il soit, reste encore « relativement simple ». Les véritables problèmes arrivent avec les

travaux de Cantor sur la cardinalité et les nombres transfinis. Cantor ouvre grand la porte sur un gouffre sans fond, qui nous emmène bien au-delà de ce que peut appréhender avec confiance l'esprit humain : l'étude de l'infini. Bien sûr, l'infini est présent en mathématique depuis l'antiquité, en premier lieu via la considération qu'il n'existe pas de plus grand nombre entier. Mais la révolution épistémologique de Cantor consiste à considérer l'infini comme un objet d'étude à part entière. Cette considération amènera aux balbutiements de ce qui deviendra un siècle plus tard la Théorie des ensembles avec un grand 'T'.

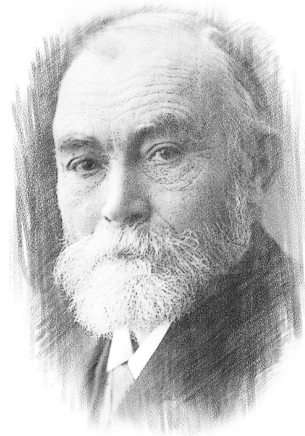
Avec les travaux de Cantor, la question de savoir *ce qu'est* l'activité mathématique s'est faite de plus en plus pressante : peut-on raisonner vraiment sur tout, et même sur l'infini, concept qui nous dépasse ? Mais si l'on accepte, comme c'est le cas aujourd'hui que l'on peut raisonner sur l'infini, on ne peut certainement pas le faire n'importe comment. Au fond, qu'est-ce *faire* des mathématiques ? En particulier quand on commence à manipuler des objets sur lesquels nous n'avons plus tellement d'intuition, comment être sûr que ce dont on parle a réellement un sens ? Ces considérations ont trouvé leur apogée lors de la fameuse « crise des fondements », qui va de la fin du XIX$^{e}$ siècle au début du XX$^{e}$. Il s'agit alors de définir des règles pour encadrer l'activité mathématique. Il s'agit en fait de définir précisément l'étude mathématique non pas d'objets comme les entiers, les réels ou encore les fonctions, mais de définir *l'étude mathématique des mathématiques elles -mêmes*. Il est *a posteriori* remarquable de constater le succès de cette entreprise : les mathématiques sont un outil suffisamment puissant pour pouvoir se définir et s'étudier elles-mêmes, avec la rigueur inhérente à la discipline ! Ce ne fut évidemment pas un chemin aisé. Dans cette entreprise, les trois mousquetaires — qui comme on le sait sont en fait au nombre de quatre — s'appellent Frege, Russel, Zermelo et Hilbert.

**Frege**

Philosophe et mathématicien allemand de la fin du XIX$^{e}$ siècle, Gottlob Frege est mu par une certitude : la logique précède les mathématiques. Mais la logique de l'époque est encore très pauvre, et se cantonne essentiellement aux travaux de George Boole sur ce que l'on appelle aujourd'hui *le calcul propositionnel* : la manipulation de propositions, vraies ou fausses, que l'on peut connecter entres elles via les « et » et « ou » logiques, bien connus des informaticiens. Pour l'ambition de Frege, ce système est trop restreint pour asseoir les mathématiques sur des fondations logiques. En particulier, rien dans le calcul propositionnel ne permet de parler d'objets spécifiques à travers les relations qu'ils entretiennent les uns avec les autres. Il formalise alors dans son ouvrage le « Begriffsschrift » un nouveau langage afin de

palier aux carences de la logique de l'époque, langage qui évoluera pour devenir ce que l'on appelle aujourd'hui *le calcul des prédicats*, omniprésent en mathématiques.

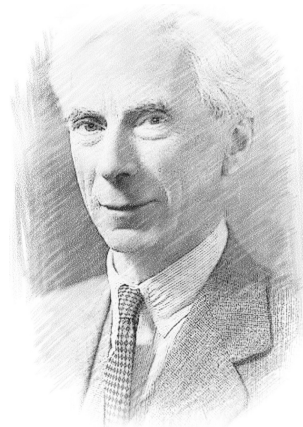
Frege est aujourd'hui considéré comme le père de la logique moderne, notamment via son concept de variables quantifiées $\forall x \ldots$, $\exists x \ldots$ Il utilise son formalisme pour s'atteler dans ses ouvrages suivants, « Fondements de l'arithmétique » (1884) et « Lois fondamentales de l'arithmétique I et II » (parus en 1893 et 1903), à fonder l'arithmétique sur la logique. Il crée pour cela une définition des entiers naturels qui peut être vue aujourd'hui comme reposant sur le concept *d'ensemble* et sur celui du *schéma d'axiomes de compréhension* : si $\Psi(x)$ est une formule mathématique qui peut être vraie ou fausse en fonction de $x$, alors l'ensemble $\{x \ : \ \Psi(x)\}$ des éléments qui satisfont cette formule est bien défini.



Gottlob Frege, 1848–1925

**Russell**

En 1902, Russell dans une lettre adressée à Frege lui fait part d'un doute concernant ses travaux. Soit $y$ l'ensemble des ensembles qui ne s'appartiennent pas : $y = \{x \ : \ x \notin x\}$. A-t-on alors $y \in y$ ou bien $y \notin y$ ? On comprend aisément la situation paradoxale à laquelle on arrive. Cet exemple restera célèbre comme étant *le paradoxe de Russell*. À 53 ans, Frege comprend alors que le paradoxe de Russell implique l'effondrement du système qu'il a mis des années à échafauder. Un coup dur dont il aura beaucoup de mal à se remettre.



Bertrand Russell, 1872–1970

Malgré ce paradoxe, Russell accueille avec beaucoup d'enthousiasme les travaux de Frege, et contribue largement à en faire reconnaître la valeur. Tout comme Frege, Russell sent le besoin d'appuyer les mathématiques sur des bases solides. Tout comme Frege,

Russell a cette intuition que la logique précède les mathématiques. Il va s'atteler dix ans durant, avec son ancien professeur Alfred Whitehead, à cette recherche de fondements logiques, qui aboutiront à leur fameux ouvrage « Principia Mathematica » : un travail titanesque qui s'étend sur plus de 2000 pages, dont l'ambition est de décrire un ensemble d'axiomes logiques et de règles d'inférence à partir desquelles toute vérité mathématique pourrait être démontrée. Ces travaux posent les fondements de ce que l'on appelle aujourd'hui *la théorie des types*, un système encore étudié aujourd'hui, présentant des liens très forts avec les langages de programmation. En parallèle, s'est développée la théorie des ensembles dont l'axiomatisation a été initiée par Zermelo, et complétée plus tard par Fraenkel et Skolem indépendamment, pour donner le système axiomatique ZF, du nom de Zermelo-Fraenkel.

**Zermelo**

Ernst Zermelo, 1871–1953

Un fait absolument remarquable est que le système ZF, auquel il est parfois nécessaire de rajouter l'axiome du choix, donne un cadre au sein duquel peuvent se formaliser *la totalité des mathématiques modernes*, si l'on exclut les récents développements de la théorie des ensembles, dont l'objectif est justement de sortir de ce système. Zermelo trouve le moyen d'éviter le fameux paradoxe de Russell — comme Russell lui-même avec sa théorie des types — en bornant l'axiome de compréhension. L'ensemble $\{x \ : \ \Psi(x)\}$ n'est désormais plus valide, il convient de partir d'un ensemble existant $y$, auquel cas on peut à présent définir l'ensemble des éléments de $y$ qui satisfont $\Psi$ : $\{x \in y \ : \ \Psi(x)\}$. Pourtant, cette théorie ne fait pas tout de suite consensus. Poincaré, s'il n'a jamais activement pris part à la crise des fondements, en a suivi les développements avec intérêt. Comme tous les protagonistes de l'époque, il est très conscient du danger qui se cache derrière le paradoxe de Russell. Il théorise même le problème sous le nom d'*imprédicativité*[1] : une définition imprédicative est en substance une définition circulaire dans laquelle l'objet qui est défini est lui-même susceptible d'être utilisé dans la définition. C'est ce qui se passe quand on définit $y = \{x \ : \ x \notin x\}$ : l'ensemble $y$ défini à gauche de l'égalité est aussi

1. Utilisé auparavant par Russell dans un sens légèrement différent.

concerné à droite du signe égal, puisque l'on considère potentiellement tous les ensembles. Si l'axiomatique de Zermelo évite le paradoxe de Russell, il reste néanmoins, indirectement imprédicatif, comme le remarquera Poincaré qui écrira [176] :

« *En posant d'avance son ensemble M* [Poincaré parle alors de la borne utilisée par Zermelo dans l'axiome de compréhension, qui permet de définir pour un ensemble $M$ existant $\{x \in M : \Psi(x)\}$], *il* [Mr. Zermelo] *a élevé un mur de clôture qui arrête les gêneurs qui pourraient venir du dehors. Mais il ne se demande pas s'il peut y avoir des gêneurs du dedans qu'il a enfermés avec lui dans son mur. Si l'ensemble M a une infinité d'éléments, cela veut dire non que ces éléments puissent être conçus comme existant d'avance tout à la fois, mais qu'il peut sans cesse en naître de nouveaux ; ils naîtront à l'intérieur du mur, au lieu de naître dehors, voilà tout.* »

Le système de Zermelo considère que si un ensemble $A$ existe, alors l'ensemble de ses parties $\mathcal{P}(A)$ existe également. Cet axiome combiné avec l'axiome de compréhension restreint permet de faire des définitions circulaires. Ainsi, dans la définition $A = \{n \in \mathbb{N} : \forall S \in \mathcal{P}(\mathbb{N})\ n \notin S\}$, le quantificateur $\forall S$ va prendre pour valeur tous les sous-ensembles de $\mathbb{N}$, et en particulier l'ensemble $A$ lui-même. L'ensemble $A$ est donc défini en fonction de lui-même. Poincaré ajoute alors à la fin de son argumentation :

« *Mais s'il* [Mr. Zermelo] *a bien fermé sa bergerie, je ne suis pas sûr qu'il n'y ait pas enfermé le loup. Je ne serais tranquille que s'il avait démontré qu'il est à l'abri de la contradiction.* »

On peut difficilement donner tort à Poincaré : comment être sûr au fond qu'un paradoxe de Russell ne surgira pas de nulle part, au détour d'une définition circulaire cachée ? Zermelo lui-même est conscient du problème, et cherchera à démontrer sans succès que son système axiomatique est *cohérent*, c'est-à-dire exempt de paradoxe. Cette recherche de preuve de la cohérence des mathématiques a connu son apogée vers 1920, sous l'impulsion de David Hilbert.

### Hilbert

Hilbert est certainement — tout comme Poincaré — un des derniers mathématiciens à avoir une connaissance approfondie de l'ensemble des mathématiques de son époque. Son œuvre est considérable, et il a profondément influencé les développements de la discipline durant le XX$^{e}$ siècle. Il participe activement à la crise des fondements en opposant à Russell une vision *formaliste* des mathématiques, plutôt qu'une vision *logiciste*. Pour Hilbert, les mathématiques doivent pouvoir se réduire à un ensemble de règles, que l'on doit pouvoir appliquer de façon purement mécanique et déconnectée de toute psychologie du mathématicien. Il imagine ainsi des systèmes de preuve, au sein desquels il différencie *les axiomes*, qui sont les

David Hilbert, 1862–1943

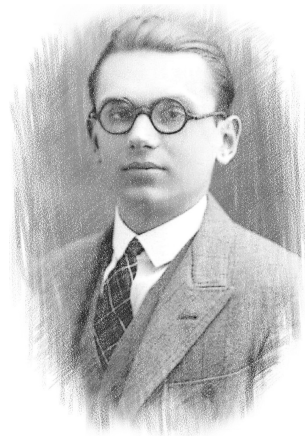
phrases mathématiques que l'on suppose vraies, par exemple les axiomes de Zermelo-Fraenkel, et les *règles de déduction*, qui permettent de combiner les axiomes entre eux pour en déduire des théorèmes. C'est la vision de Hilbert qui finira par s'imposer, même si cela ne se sera pas accompli sans remous. L'objectif ultime pour Hilbert est de montrer via des systèmes de déduction considérés comme sûr — en particulier via des raisonnements finis sur des objets finis — que l'ensemble des mathématiques, qui elles font appel à des objets infinis dont la pertinence est sujette à caution, forme un système cohérent, c'est-à-dire exempt de paradoxe : c'est ce que l'on appellera le *programme de Hilbert*, dont le Entscheindungsproblem évoqué dans la section 6-1 constitue l'un des aspects. Ce programme connaîtra un coup d'arrêt brutal avec les travaux de Gödel, qui démontre dix ans plus tard son fameux théorème d'incomplétude : l'arithmétique elle-même est impuissante à démontrer qu'elle est exempte de paradoxe.

### Le cinquième mousquetaire

Les travaux de Gödel apportent une conclusion aussi magistrale qu'inattendue à la crise des fondements. Gödel montre deux choses : même les systèmes les plus simples et les mieux compris, comme l'arithmétique, qui ne parle que d'objets finis, contiennent des vérités indémontrables. En particulier, et à supposer qu'il soit vrai que les axiomes de l'arithmétique forment un système exempt de paradoxes, alors la cohérence de l'arithmétique est elle-même l'une de ces vérités indémontrables. Gödel montre enfin — avec l'aide de Rosser — que l'ajout d'axiomes ne change rien à l'affaire : n'importe quel système d'axiomes cohérent, contenant l'arithmétique, et « dont on peut connaître les axiomes » ne peut démontrer sa propre cohérence. Gödel développe pour cela les premières versions de ce qui sera plus tard la calculabilité : « dont on peut connaître les axiomes » signifie calculable, dans un sens similaire au sens moderne.

Le retentissement dans le monde mathématique est colossal. Le programme de Hilbert est à terre, et les mathématiques n'auront jamais de fondations entièrement satisfaisantes.

Encore aujourd'hui, on ne sait pas si le système ZF qui axiomatise l'ensemble des mathématiques est cohérent : et pour cause, si comme on l'espère il est bien exempt de paradoxe, on ne pourra jamais le démontrer mathématiquement, ou en tout cas tant que l'on se cantonne aux axiomes de ZF. L'impact épistémologique est considérable. Les mathématiques, mère des sciences exactes, sont non seulement tributaires d'une *croyance*, mais sont en plus capables de démontrer qu'il en sera toujours ainsi !



Kurt Gödel, 1906–1978

# 2. La logique du premier ordre

De Frege et Russell on retiendra le langage logique moderne utilisé en mathématiques, de Hilbert on retiendra un système de preuve à base d'axiomes et de règles de déductions applicables aux énoncés mathématiques, et de Zermelo on retiendra le système axiomatique ZF ou ZFC, suffisant pour formaliser l'ensemble des mathématiques traditionnelles. Nous présentons à présent sans rentrer dans trop de détails les principes de base de la logique du premier ordre, notre objectif étant de présenter de manière un peu plus précise le théorème de Gödel et ses conséquences. Pour cette raison, le fil rouge de notre présentation sera l'exemple spécifique de l'arithmétique de Peano.

## 2.1. Langage de l'arithmétique

La première étape pour formaliser nos démonstrations mathématiques est de fixer le langage utilisé. Nous allons pour cela définir le langage de l'arithmétique de Peano.

**Définition 2.1.** Le langage $\mathcal{L}_{\mathrm{PA}}$ de l'arithmétique de Peano comprend les symboles spécifiques au calcul des prédicats :

(1) des symboles de *variables* $x, y, z, \ldots$ : elles représentent des entiers naturels ;

(2) les parenthèses ( ) et les symboles de *connecteurs logiques* : $\wedge, \vee, \to, \neg$ ;

(3) des symboles de *quantificateurs* : $\forall, \exists$.

Et ceux spécifiques à l'arithmétique de Peano :

(1) les symboles de *fonctions binaires* suivantes : $+$, $\times$ ;

(2) les symboles de *relations binaires* suivantes : $=, <$ ;

(3) des symboles de *constantes* $\dot{0}, \dot{1}$. ♢

Un langage n'est rien d'autre qu'une liste de symboles, mais ces symboles ont vocation à être utilisés avec un sens précis. En ce qui concerne le calcul des prédicats, il s'agit du sens usuel : par exemple, $\ll \wedge \gg$ est le *et* logique ou encore $\ll \exists \gg$ est la quantification existentielle. En ce qui concerne les symboles spécifiques à l'arithmétique de Peano, il y a d'abord les fonctions $+$ et $\times$ qui représentent respectivement l'addition et la multiplication, les symboles d'égalité et d'inégalité qui ont leurs sens usuels sur les entiers, et enfin les constantes $\dot{0}, \dot{1}$ qui chacune représente l'entier respectif correspondant.

**Langages du premier ordre**

On voit sans peine comment généraliser la définition précédente pour obtenir d'autres langages. Les symboles spécifiques au calcul des prédicats sont les mêmes dans tous les langages du premier ordre, auxquels on ajoute un nombre arbitraire de symboles de fonctions ($n$-aire pour des entiers $n$ arbitraires), un nombre arbitraire de symboles de relations (également $n$-aire pour des entiers $n$ arbitraires) et un nombre arbitraire de symboles de constantes.

Les symboles de fonctions sont soumis à des règles d'agencement pour former ce que l'on appelle les *termes* du langage.

**Définition 2.2.** Les *termes* du calcul des prédicats de l'arithmétique sont définis inductivement de la manière suivante.

(1) Une variable ou une constante est un terme.

(2) Si $t_1, t_2$ sont des termes, alors $(t_1 + t_2)$ et $(t_1 \times t_2)$ sont des termes. ♢

**Exemple 2.3.** Les expressions suivantes sont des termes :

$$x, \quad ((((x + \dot{1}) + \dot{1}) + \dot{1}) + \dot{1} + \dot{1}), \quad (\dot{1} + \dot{0}), \quad (x + (y \times z)).$$

Comme nous pouvons le voir, le langage de l'arithmétique est assez minimaliste, et les expressions valides sont très structurées pour ôter toute ambiguïté.

En pratique, on utilisera un certain nombre de raccourcis de notation pour améliorer la lisibilité, tant que la traduction en termes valides est non ambigüe. Par exemple, $t_0 + t_1 + t_2$ est un raccourci pour $((t_0 + t_1) + t_2)$. De même, $x + \dot{3}$ est un raccourci pour $(x + ((\dot{1} + \dot{1}) + \dot{1}))$.

**Définition 2.4.** Un terme est *clos* s'il ne contient aucune variable, et donc uniquement des constantes et les opérations $+$ et $\times$. ♢

Intuitivement, un terme clos de l'arithmétique est une manière de représenter un entier naturel. Par exemple, $(\dot{1} + \dot{1}) \times \dot{0}$ est un nom pour l'entier 0. Les entiers possèdent chacun une infinité de noms.

**Exemple 2.5.** Le terme $x + \dot{1}$ n'est pas clos, contrairement à $(\dot{1} + \dot{1}) \times \dot{0}$.

Si les symboles de fonctions sont utilisés pour créer les termes du langage, les symboles de relations sont eux utilisés pour créer les *formules* du langage.

**Définition 2.6.** Les *formules de l'arithmétique* sont définies comme suit.

(1) Pour tous termes $t_1, t_2$, alors $t_1 = t_2$ et $t_1 < t_2$ sont des formules. Ces formules sont appelées *formules atomiques*. Les formules atomiques auxquelles on ajoute leurs négations, ici, $\neg t_1 = t_2$ et $\neg t_1 < t_2$, sont appelées *littéraux*.

(2) Pour toutes formules $F_1, F_2$, alors $(F_1 \wedge F_2), (F_1 \vee F_2), (F_1 \to F_2)$ et $\neg F_1$ sont des formules.

(3) Pour toute formule $F$, alors $\forall x F$ et $\exists x F$ sont des formules. ♢

Là encore, on aura recours au sucre syntaxique en notant $t_1 \leqslant t_2$ la formule $(t_1 < t_2) \vee (t_1 = t_2)$ et $F_1 \leftrightarrow F_2$ la formule $(F_1 \to F_2) \wedge (F_2 \to F_1)$.

**Formules de premier ordre**

Ici aussi, on généralise sans peine la formation de formules et de termes dans n'importe quel langage : les symboles de fonction du langage servent à créer les termes, qui servent ensuite avec l'aide des symboles de relation à créer des formules atomiques, qui peuvent ensuite être composées entre elles avec l'aide des symboles du calcul des prédicats comme dans (2) et (3) de la définition précédente.

Avec les quantificateurs apparaissent les notions de variables *libres* et de variables *liées* : les variables *liées* sont sans surprise celles qui sont liées à un quantificateur, et les variables libres sont celles qui ne le sont pas. La définition formelle est assez lourde, mais quelques exemples suffisent à s'en créer une intuition.

**Exemple 2.7.** Dans la formule « $\forall x\ \exists y\ y = x + \dot{1}$ », les variables $x$ et $y$ sont liées, alors que dans la formule « $\exists y\ y = x + 1$ » seule la variable $y$ est liée, contrairement à la variable $x$ qui est libre.

**Définition 2.8.** Une *formule close* ou un *énoncé* est une formule dans laquelle aucune variable n'est libre. ♢

**Notation**

Étant donné une formule $F$ ayant pour variables libres $x_1, \ldots, x_n$, on écrira $F(x_1, \ldots, x_n)$ pour signifier que les variables libres de $F$ sont $x_1, \ldots, x_n$.

Intuitivement, une formule close est une affirmation qui aura une valeur de vérité (vraie ou fausse) une fois évaluée sur les entiers. Les formules possédant des variables libres définissent quant à elles des prédicats sur les entiers.

### 2.2. Les systèmes de déduction à la Hilbert

Afin de formaliser mathématiquement la notion de démonstration, Hilbert a imaginé un système de règles bien précises qui suffisent à montrer « tout ce qui est démontrable ». Comment le sait-on ? Cette idée sera rendue précise avec le théorème de complétude de Gödel à venir. Par la suite, de nombreux autres systèmes de démonstration ont été développés, tous équivalents et plus ou moins adaptés à certains objectifs.

#### 2.2.1. Axiomes et règles

Dans un système à la Hilbert, une démonstration est une liste finie de phrases mathématiques $F_0, F_1, F_2, \ldots, F_n$ — des formules dans le langage considéré — satisfaisant les règles suivantes : pour tout $i \leqslant n$, soit $F_i$ est un axiome, soit $F_i$ est produit à partir de règles d'inférence appliquées à des formules $F_{j_1}, \ldots, F_{j_m}$ pour $j_1, \ldots, j_m < i$. Chaque phrase $F_i$ dans cette liste sera alors démontrée, l'objectif étant normalement d'obtenir $F_n$, la dernière d'entre elles. Voyons à présent un exemple précis de système à la Hilbert suffisamment puissant pour démontrer tout ce qui est démontrable.

**Les axiomes** : les axiomes que nous pouvons toujours utiliser sont les tautologies de la logique du premier ordre. Ainsi, par exemple $A \vee \neg A$ pourra être utilisé comme axiome. On en distingue trois types.

1. Les tautologies de la logique propositionnelle. Par exemple, $(F \to G) \to (\neg G \to \neg F)$ est une tautologie de la logique propositionnelle : elle sera vraie pour n'importe quelle formule $F$ ou $G$ indépendamment de leur valeur de vérité.
2. Les tautologies du calcul des prédicats. En pratique, seuls quatre schémas d'axiomes sont nécessaires :
   (a) $\forall x(F \to G) \to (F \to \forall x G)$ pour toute formule $F$ ne contenant pas la variable $x$, et toute formule $G$ ;

(b) $\exists x(F \to G) \to (\exists x F \to G)$ pour toute formule $F$, et toute formule $G$ ne contenant pas la variable $x$ ;

(c) $\forall x F \to F_{t/x}$ pour tout terme $t$ et toute formule $F$ ne contenant aucune variable de $t$ ;

(d) $F_{t/x} \to \exists x F$ pour tout terme $t$ et toute formule $F$ ne contenant aucune variable de $t$.

Ci-dessus, $F_{t/x}$ désigne la formule $F$ pour laquelle chaque occurrence de $x$ est remplacée par le terme $t$.

3. Les axiomes de l'égalité :

(e) $t = t$ pour tout terme $t$ ;

(f) $t_1 = q_1 \wedge \cdots \wedge t_n = q_n \to f(t_1, \ldots, t_n) = f(q_1, \ldots, q_n)$ pour tout $n$, tous termes $(t_i)_{1 \leqslant i \leqslant n}, (q_i)_{1 \leqslant i \leqslant n}$ et tout symbole de fonction $n$-aire $f$ ;

(g) $t = q \to (F(t/z) \to F(q/z))$ pour tous termes $t, q$ et toute formule $F(z)$ ne faisant pas intervenir des variables de $t$ ou $q$.

Ci-dessus, $F(t/z)$ et $F(q/z)$ désignent la formule $F$ dans laquelle chaque occurrence de $z$ est remplacée respectivement par $t$ et $q$.

Notons que ces schémas d'axiomes dépendent du langage considéré, chaque langage utilisant des symboles de fonctions et relations qui leur sont spécifiques pour construire respectivement les termes et les formules atomiques.

**Symbole d'égalité**

Nous considérons ici que le symbole d'égalité fait nécessairement partie du langage que l'on utilise, et aura toujours son sens usuel, ce qui justifie les axiomes de l'égalité mentionnés ci-dessus.

**Les règles d'inférence.** Les règles d'inférence permettent de combiner des phrases déjà démontrées dans notre liste, pour en obtenir de nouvelles. Les deux règles suivantes sont suffisantes.

1. *Règle 1 : le Modus Ponens*, à la base de tout raisonnement déductif. Si $A \to B$ est démontré et si $A$ est démontré, alors on peut en déduire $B$.
2. *Règle 2 : la généralisation.* Si $F(x)$ est démontré pour une variable $x$ libre dans $F$, on peut alors en déduire $\forall x F(x)$. Cette règle est très utilisée en mathématiques : si l'on veut prouver par exemple que pour tous rationnels $x < y$, il existe un rationnel $z$ tel que $x < z < y$, on commence par fixer des variables rationnelles $x, y$ sur lesquelles on ne suppose rien d'autre que $x < y$. Si l'on arrive à déduire l'existence d'un rationnel $z$ tel que $x < z < y$, sans utiliser aucune propriété spécifique de $x, y$, on en déduit par la règle de généralisation que pour tous rationnels $x < y$, il existe un rationnel $z$ tel que $x < z < y$.

Cela conclut la description de notre système à la Hilbert. Voyons tout de suite un exemple de démonstration.

**Exemple 2.9.** Montrons $\forall x F(x) \to \exists x F(x)$. Pour plus de lisibilité, on notera $A \equiv \forall x F(x)$, $B \equiv F(y)$ et $C \equiv \exists x F(x)$.

(1) $A \to B$ (axiome (c)).

(2) $B \to C$ (axiome (d)).

(3) $(A \to B) \to ((B \to C) \to ((A \to B) \wedge (B \to C)))$ (tautologie).

(4) $(B \to C) \to ((A \to B) \wedge (B \to C))$ (Modus Ponens sur (1) et (3)).

(5) $(A \to B) \wedge (B \to C)$ (Modus Ponens sur (2) et (4)).

(6) $((A \to B) \wedge (B \to C)) \to (A \to C)$ (tautologie).

(7) $A \to C$ (Modus Ponens sur (5) et (6)).

**Quid des univers vides ?**

Le lecteur pourra être surpris par la phrase $\forall x F(x) \to \exists x F(x)$ — que l'on a démontrée. Que se passe-t-il si l'on se place dans un univers vide ? À ce moment, $\forall x F(x)$ est bien vérifié, mais pas $\exists x F(x)$. L'axiome (d) ci-dessus implique effectivement que les formules démontrées ne seront valides que s'il existe au moins un élément dans notre univers. La notion d'univers sera rendue précise dans la section 2.4 à venir. Il est de fait nécessaire d'avoir une telle restriction si l'on veut démontrer que toute formule est équivalente à une formule sous forme prénexe (voir la définition 2.12).

### 2.2.2. Premiers outils

Nous prétendons que le système à la Hilbert décrit ci-dessus permet de montrer tout ce qui est démontrable, et nous le verrons formellement avec le théorème de complétude à venir. Le système est volontairement minimaliste, et difficile à manipuler tel quel. Le lecteur pourra par exemple essayer de démontrer $\neg\forall x F(x) \to \exists x\, \neg F(x)$ pour se rendre compte de la difficulté d'utilisation du système en l'état. Le mathématicien qui veut faire ce genre de preuve procèdera naturellement comme pour toute démonstration : en supposant que $\neg\forall x F(x)$ est vrai, et en essayant d'en déduire $\exists x\, \neg F(x)$. Le problème est que si l'on veut respecter le formalisme d'un système à la Hilbert, $\neg\forall x F(x)$ n'est pas nécessairement un axiome que l'on peut supposer vrai afin d'en dériver une conclusion. Nous voyons alors notre premier outil fondamental, qui va nous permettre de procéder comme on en a l'habitude.

**Lemme 2.10 (Lemme de déduction).** Soit $F$ une formule close. Si l'on peut faire une démonstration de $G$ en utilisant $F$ comme axiome, il existe alors une démonstration de $F \to G$ (qui n'utilise pas $F$ comme axiome). ⋆

PREUVE. Soit $G_1, \ldots, G_n$ une démonstration de $G_n$ utilisant $F$ comme axiome. Montrons par récurrence sur la taille d'une démonstration que l'on peut faire quelques insertions dans la suite $F \to G_1, \ldots, F \to G_n$ afin d'en faire une démonstration valide n'utilisant pas $F$ comme axiome.
Si $G_i$ est l'énoncé $F$, alors $F \to F$ est un axiome de la logique propositionnelle. Si $G_i$ est un axiome de la logique propositionnelle, alors c'est aussi le cas de $F \to G_i$. Si $G_i$ est l'un des axiomes (a), (b), (c), (d) du calcul des prédicats, alors $G_i \to (F \to G_i)$ est un axiome de la logique propositionnelle. En utilisant le Modus Ponens sur $G_i$ et $G_i \to (F \to G_i)$, on obtient bien $F \to G_i$. Si $G_i = \forall x\ G_j$ pour $j < i$ est obtenu par la règle de généralisation, alors $\forall x\ (F \to G_j)$ est obtenu à partir de $F \to G_j$ (que l'on a par hypothèse de récurrence) par la règle de généralisation. Par l'axiome (a), on a

$$\forall x(F \to G_j) \to (F \to \forall x\ G_j),$$

de sorte que par Modus Ponens on obtient $F \to \forall x G_j$. Enfin, si $G_i = G_b$ est obtenu par Modus Ponens sur $G_a$, alors $G_a \to G_b$ pour $a, b < i$. Alors, on a $F \to G_a$ et $F \to (G_a \to G_b)$, par hypothèse de récurrence. La formule

$$\big(F \to (G_a \to G_b)\big) \ \to\ \big((F \to G_a) \to (F \to G_b)\big)$$

est une tautologie du calcul des prédicats.
Par Modus Ponens, on en déduit $(F \to G_a) \to (F \to G_b)$ et, par une deuxième application du Modus Ponens, on en déduit $F \to G_b$. ∎

Voyons tout de suite un exemple d'application du lemme de déduction pour démontrer $\neg\forall x F(x) \to \exists x\ \neg F(x)$.

**Exemple 2.11.** Montrons $\neg\exists x F(x) \to \forall x\ \neg F(x)$. D'après le lemme de déduction, on peut supposer $\neg\exists x F(x)$ comme axiome.

(1) $\neg\exists x F(x)$ (axiome).

(2) $F(x) \to \exists x F(x)$ (axiome (d)).

(3) $(F(x) \to \exists x F(x)) \to (\neg\exists x F(x) \to \neg F(x))$ (tautologie).

(4) $\neg\exists x F(x) \to \neg F(x)$ (Modus Ponens sur (2) et (3)).

(5) $\neg F(x)$ (Modus Ponens sur (1) et (4)).

(6) $\forall x\ \neg F(x)$ (généralisation sur (5)).

Montrons à présent $\forall x\ \neg\neg F(x) \to \forall x\ F(x)$.

(1) $\forall x\ \neg\neg F(x)$ (axiome).

(2) $\forall x\ \neg\neg F(x) \to \neg\neg F(x)$ (axiome (c)).

(3) $\neg\neg F(x)$ (Modus Ponens sur (1) et (2)).

(4) $\neg\neg F(x) \to F(x)$ (tautologie).

(5) $F(x)$ (Modus Ponens sur (3) et (4)).

(6) $\forall x F(x)$ (généralisation sur (5)).

On laisse au lecteur le soin d'utiliser la contraposée pour en déduire

$$\neg\forall x F(x) \to \exists x \neg F(x).$$

### 2.2.3. Forme prénexe

Ce système de démonstration nous permet d'établir que toute formule — dans un langage quelconque— est prouvablement équivalente à une formule sous *forme prénexe*.

**Définition 2.12.** Une formule est sous *forme prénexe* si elle est de la forme $Q_1 x_1 \dots Q_n x_n \ F(x_1, \dots, x_n, y_1, \dots y_m)$ où chaque $Q_i$ est un quantificateur $\forall$ ou $\exists$ et $F(x_1, \dots, x_n, y_1, \dots y_m)$ est une formule sans quantificateur. ♢

On laisse au lecteur le soin de montrer les équivalences suivantes :

- ▷ $\forall x F \ \wedge \ G \equiv \forall x (F \wedge G)$ ;
- ▷ $\forall x F \ \vee \ G \equiv \forall x (F \vee G)$ ;
- ▷ $\exists x F \ \wedge \ G \equiv \exists x (F \wedge G)$ ;
- ▷ $\exists x F \ \vee \ G \equiv \exists x (F \vee G)$.

Chacune de ces équivalences, couplées à l'exemple 2.11, permet de transformer n'importe quelle formule en une formule prénexe prouvablement équivalente dans notre système de déduction, en déplaçant petit à petit les quantificateurs vers la gauche.

Notons que les équivalences ci-dessus ne tiennent que si l'on se place dans un univers possédant au moins un élément. Ainsi, par exemple, aura-t-on $(\forall x \ x = x) \wedge (\exists y \ y \neq y)$ manifestement faux dans un univers vide, alors que $\forall x \ (x = x \wedge (\exists y \ y \neq y))$ est toujours vrai.

## 2.3. Théories logiques et arithmétique de Peano

Une fois un langage fixé — dans notre cas celui de l'arithmétique — et le système de démonstration spécifié, on peut alors considérer une *théorie mathématique* dans ce langage, et l'utiliser pour démontrer des théorèmes concernant la structure décrite par cette théorie.

**Définition 2.13.** Une *théorie* $T$ dans un langage $\mathcal{L}$ est une collection de formules closes de ce langage. On utilise aussi souvent le terme *système axiomatique* ou plus simplement *système* pour désigner une théorie. ♢

La théorie est alors vue comme une liste d'axiomes, que l'on peut utiliser dans nos démonstrations, en plus des axiomes présents dans le système de démonstration.

Voyons tout de suite les axiomes de l'arithmétique qui furent mis au point par Peano vers la fin du XIX$^{e}$ siècle.

### 2.3.1. Axiomes de l'arithmétique de Peano

Les axiomes de Peano permettent de spécifier le comportement des entiers naturels. La première série d'axiomes définit le comportement des entiers vis-à-vis du successeur.

(1) $\forall x\ \neg(x + \dot{1} = \dot{0})$ : 0 n'a pas de prédécesseur.

(2) $\forall x\ (x = \dot{0} \vee \exists y\ (x = y + \dot{1}))$ : tout entier différent de 0 a un prédécesseur.

(3) $\forall x\ \forall y\ (x + \dot{1} = y + \dot{1} \rightarrow x = y)$ : la fonction successeur pour les entiers est injective.

Les axiomes suivants donnent des règles pour calculer l'addition et la multiplication :

(4) $\forall x\ (x + \dot{0} = x)$ ;

(5) $\forall x \forall y\ (x + (y + \dot{1}) = (x + y) + \dot{1})$ ;

(6) $\forall x\ (x \times \dot{0} = \dot{0})$ ;

(7) $\forall x \forall y\ (x \times (y + \dot{1}) = (x \times y) + x)$.

Enfin, on définit le comportement des entiers vis-à-vis de l'ordre :

(8) $\forall x \forall y\ (x < y \leftrightarrow (\exists z\ (z \neq \dot{0} \wedge x + z = y)))$.

**Notation**

On note $\mathsf{Q}$ la théorie composée des axiomes (1)-(8), qui forment ce que l'on appelle l'*arithmétique de Robinson*.

Pour obtenir l'arithmétique de Peano, on ajoute l'axiome suivant, pour toute formule de l'arithmétique $F(x)$ :

(9) $\Big(F(0) \wedge \big(\forall x\big(F(x) \rightarrow F(x + \dot{1})\big)\big)\Big) \rightarrow \forall x F(x)$

Notons que l'axiome (9) n'est pas un unique axiome. Tout comme pour les axiomes (a), (b), (c), (d) de notre système de démonstration, il s'agit d'un *schéma d'axiomes*, c'est-à-dire d'une infinité d'axiomes paramétrés par une formule, ici $F(x)$.

L'énoncé (9) est l'axiome bien connu de l'induction sur les entiers : si une formule $F$ est vraie pour l'entier 0, et si le fait qu'elle soit vraie pour $n$ implique qu'elle le soit pour $n + 1$, alors elle est vraie pour tout entier $n$.

**Notation**

On note PA la théorie composée de $\mathsf{Q}$ et du schéma d'axiome (9) pour toute formule de l'arithmétique. C'est cette théorie que l'on appelle *arithmétique de Peano*.

Nous verrons dans le chapitre 23 comment utiliser les axiomes de PA pour montrer quelques faits élémentaires sur les entiers naturels. Nous verrons en particulier que le schéma d'induction est équivalent au schéma suivant : pour toute formule $F$ vraie pour au moins un entier, il existe un plus petit entier $x$ tel que $F(x)$ est vrai. Cela peut bien entendu sembler parfaitement évident, car nous avons en tête la structure des entiers naturels $\mathbb{N}$ que nous connaissons bien, mais une démonstration n'utilise pas cette structure : elle utilise uniquement les axiomes, et dans le cas de l'arithmétique, ceux-ci sont précisément faits pour que toute structure mathématique les vérifiant se comporte comme les entiers naturels. Cela nous amènera à la notion de *modèle*, dans la section suivante.

### 2.3.2. Démonstrations dans une théorie

Une fois que l'on a fixé une théorie, on peut en utiliser les axiomes au sein d'un système à la Hilbert pour démontrer des énoncés mathématiques.

**Notation**

On écrira $T \vdash F$ pour signifier qu'il existe une démonstration de la formule $F$ à partir des axiomes de $T$ via le système à la Hilbert exposé dans la section 2.2. S'il n'existe pas de telle démonstration, on écrira alors $T \nvdash F$.

Parmi les tautologies de la logique propositionnelle, on trouve pour toutes formules $F, G$ la formule $(F \wedge \neg F) \to G$, appelée « ex falso quodlibet », signifiant qu'à partir d'une contradiction $(F \wedge \neg F)$ on peut déduire ce que l'on veut. Il s'ensuit que si une théorie permet de prouver une formule et son contraire, tout énoncé est prouvable dans cette théorie, ce qui lui retire tout intérêt. On attendra donc avant tout d'une théorie qu'elle soit *cohérente*.

**Définition 2.14.** Une théorie $T$ est *cohérente* s'il n'existe aucune formule $F$ telle que $T \vdash F \wedge \neg F$. ♢

**Notation**

On écrira $T \vdash \bot$ pour signifier $T \vdash F \wedge \neg F$ pour une certaine formule $F$, la notation $T \nvdash \bot$ signifiant alors logiquement que pour toute formule $F$ on a $T \nvdash F \wedge \neg F$. Comme le symbole d'égalité fait partie de notre langage, on peut considérer sans perte de généralité $\bot$ comme étant $\neg x = x$. Comme $x = x$ est un axiome, si $T \vdash \neg x = x$, alors $T \vdash x = x \wedge \neg x = x$.

Nous avons vu dans l'introduction de notre interlude que l'incohérence trouvée par Russell dans le travail de Frege fut une pierre angulaire dans la crise des fondements. Aussi les mathématiciens aimeraient-ils autant que possible être certains de ne travailler qu'avec des théories cohérentes. Mais

comment peut-on vérifier la cohérence d'une théorie ? Sans même parler de théorie, qu'en est-il de la cohérence du système à la Hilbert lui-même et de ses axiomes de la logique, présentés dans la section 2.2 ? La notion de *modèle* permet de résoudre ces questions.

### 2.4. Structures, modèles et théorème de cohérence

La notion de structure peut se définir de manière très générale pour tout langage fixé. Commençons par l'exemple qui nous intéresse plus spécifiquement, à savoir le langage de l'arithmétique.

**Définition 2.15.** Une *structure* $\mathcal{M} = (M, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, =^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$ dans $\mathcal{L}_{\mathrm{PA}}$ est donnée par :

- ▷ nn ensemble non vide $M$ ;
- ▷ des fonctions $+^{\mathcal{M}}, \times^{\mathcal{M}} : M \times M \to M$ correspondant aux symboles de fonction $+, \times$ ;
- ▷ une relation $<^{\mathcal{M}} \subseteq M \times M$ correspondant au symbole de relation $<$ ;
- ▷ une relation $=^{\mathcal{M}} \subseteq M \times M$ correspondant au symbole de relation $=$, et qui correspond à « la vraie égalité », c'est-à dire telle que
$$(x, y) \in\ =^{\mathcal{M}} \leftrightarrow x = y.$$
- ▷ Des éléments $0^{\mathcal{M}}, 1^{\mathcal{M}} \in \mathcal{M}$ correspondant respectivement aux symboles de constante $\dot{0}$ et $\dot{1}$. ♢

Une structure est donc un ensemble, ainsi que des fonctions, relations et constantes sur cet ensemble, constituant une interprétation des symboles du langage.

Par abus de notation, on confondra parfois $\mathcal{M}$ avec son *ensemble sous-jacent* $M$ (on notant par exemple $x \in \mathcal{M}$). Pour simplifier les notations, on supprimera parfois les exposants $^{\mathcal{M}}$ lorsqu'il sera clair que l'on parle des fonctions et relations de la structure et non de symboles du langage.

**Structure du premier ordre**

On voit sans peine comment généraliser la définition précédente pour obtenir des structures pour tout langage du premier ordre. On a toujours un ensemble $M$ non vide. Chaque symbole de fonction $n$-aire $f$ correspond à une fonction de $f^{\mathcal{M}} : M^n \to M$, chaque symbole de relation $n$-aire $R$ correspond à une relation $R^{\mathcal{M}} \subseteq M^n$ et chaque symbole de constante $c$ correspond à une constante $c^{\mathcal{M}} \in M$. La relation d'égalité sera toujours présente et correspondra toujours à « la vraie égalité ».

Une formule, comme par exemple

$$F(x) = \exists y\ y \times (\dot{1} + \dot{1}) = x,$$

n'est qu'une suite de symboles. Une fois fixée une structure $\mathcal{M}$, chaque symbole a vocation à être interprété par l'objet qui lui correspond dans $\mathcal{M}$ ; de plus, les variables libres pourront également être remplacées par divers *paramètres* — c'est-à-dire divers éléments — de la structure.

**Définition 2.16 (Formules paramétrées)**
Soit $\mathcal{L}$ un langage, et soit $\mathcal{M}$ une structure pour $\mathcal{L}$. Étant donné une formule $F(x_1,\dots,x_n)$ de $\mathcal{L}$ ayant comme variables libres $x_1,\dots,x_n$, et étant donné $a_1,\dots,a_n \in \mathcal{M}$, l'expression $F(a_1,\dots,a_n)$ désigne une *formule paramétrée* par $a_1,\dots,a_n$ : il s'agit de la formule $F$ au sein de laquelle chaque occurrence libre de $x_i$ est remplacée par $a_i$ pour $1 \leqslant i \leqslant n$. Une formule paramétrée sans variable libre sera une *formule paramétrée close*. ♢

Une formule paramétrée close n'est plus une simple suite de symboles, mais un énoncé qui sera *vrai* ou *faux* dans la structure considérée.

**Remarque**

Notons que la notion de formule paramétrée induit celle de termes paramétrés. Ainsi, dans la structure usuelle des entiers pour le langage de l'arithmétique, $(5+4)\times 2$ sera un terme paramétré par les éléments $5, 4$ et $2$ de notre structure.

Nous définissons à présent la satisfaction dans une structure, pour les formules closes ou bien paramétrées dans cette structure.

**Définition 2.17.** Soit $\mathcal{L}$ un langage, et soit $\mathcal{M} = (M, \dots)$ une structure pour $\mathcal{L}$. On dit qu'une formule $F(x_1,\dots,x_n)$ de $\mathcal{L}$ est *vraie* dans $\mathcal{M}$ pour des paramètres $a_1,\dots,a_n \in \mathcal{M}$, et l'on note $\mathcal{M} \models F(a_1,\dots,a_n)$, si $F(a_1,\dots,a_n)$ est de fait vérifiée dans la structure. La définition se fait formellement par induction sur les formules (dans ce qui suit, $\overline{x}$ et $\overline{a}$ sont des raccourcis pour $x_1,\dots,x_n$ et $a_1,\dots,a_n$).

- ▷ Cas de base : $\mathcal{M} \models R(t_1(\overline{a}),\dots,t_m(\overline{a}))$, où $R$ est un symbole de relation $m$-aire du langage correspondant à la relation $R^{\mathcal{M}} \subseteq M^m$ et chaque $t_i(\overline{x})$ est un terme, ssi $(t_1^{\mathcal{M}}(\overline{a}),\dots,t_n^{\mathcal{M}}(\overline{a})) \in R^{\mathcal{M}}$, où chaque $t_i^{\mathcal{M}}(\overline{a})$ est l'élément de $M$ obtenu en appliquant les fonctions correspondant à chaque symbole de fonction utilisé dans $t_1(\overline{a})$.
- ▷ Quantification universelle : $\mathcal{M} \models \forall y\ G(y,\overline{a})$ ssi $\mathcal{M} \models G(b,\overline{a})$ pour tout $b \in M$.

- ▷ Quantification existentielle : $\mathcal{M} \models \exists y\ G(y, \overline{a})$ ssi il existe $b \in M$ tel que $\mathcal{M} \models G(b, \overline{a})$.
- ▷ Négation : $\mathcal{M} \models \neg G(\overline{a})$ ssi $\mathcal{M} \not\models G(\overline{a})$.
- ▷ Conjonction : $\mathcal{M} \models G_1(\overline{a}) \wedge G_2(\overline{a})$ ssi $\mathcal{M} \models G_1(\overline{a})$ et $\mathcal{M} \models G_2(\overline{a})$.
- ▷ Disjonction : $\mathcal{M} \models G_1(\overline{a}) \vee G_2(\overline{a})$ ssi $\mathcal{M} \models G_1(\overline{a})$ ou $\mathcal{M} \models G_2(\overline{a})$.

La satisfaction de l'implication se déduit de celle de la négation et de la disjonction. ♢

Une fois fixée une théorie, on peut considérer des modèles de cette théorie, c'est-à-dire des structures au sein desquelles chaque axiome de la théorie sera vrai.

**Définition 2.18.** Soit $T$ une théorie dans un langage $\mathcal{L}$. Une structure $\mathcal{M}$ dans le langage $\mathcal{L}$ est un *modèle* de $T$ si chaque axiome de $T$ est vrai dans $\mathcal{M}$. ♢

**Exemple 2.19.**
- ▷ L'ensemble $\mathbb{Z}$ muni des opérations usuelles n'est pas un modèle de l'arithmétique de Peano, car il ne satisfait pas l'axiome (1) : 0 n'a pas de prédécesseur.
- ▷ L'ensemble $2\mathbb{N}$ des entiers pairs, où $\dot{0}$ est interprété par 0 et $\dot{1}$ est interprété par 2, n'est pas non plus un modèle de l'arithmétique de Peano, car il ne vérifie pas l'axiome (7) : $2 \times (2 + 2) \neq (2 \times 2) + 2$.
- ▷ Le modèle par excellence de l'arithmétique de Peano est bien entendu celui des entiers naturels $\mathbb{N}$, munis des opérations et relations usuelles.

Les théories forment l'aspect *syntaxique* des mathématiques, les modèles en forment eux l'aspect *sémantique*. Les avantages à réfléchir sur les modèles d'une théorie sont multiples.

En premier lieu, les modèles sont ce sur quoi ont naturellement travaillé les mathématiciens depuis les origines. Aujourd'hui, les mathématiques sont tellement avancées dans l'abstraction que c'est un aspect des choses que l'on perd parfois de vue, mais cette science n'est au départ pas si éloignée que cela de la physique, en ce sens qu'il s'agit d'abord d'un travail *d'observation* de la réalité de certains phénomènes afin d'en extraire les lois logiques qui les régissent. Chaque mathématicien sait par expérience qu'il ne décide pas de la vérité, celle-ci réside parfois bien cachée dans les structures abstraites étudiées, autrement dit dans les modèles. Cette méthodologie d'observation et de recherche basée sur la logique donne son caractère universel et transcendant à la vérité mathématique, constituant en quelque sorte le ciment qui lie la communauté des mathématiciens. Si la syntaxe est bien

sûr importante, car elle constitue le langage permettant de transmettre les mathématiques, *la sémantique précède cette syntaxe*[2] : c'est de là que partent les intuitions et sur elle que portent les théorèmes.

Il y a enfin un avantage plus prosaïque à l'étude des modèles : par essence, si une formule close $F$ est vraie dans un modèle, alors sa négation $\neg F$ ne peut pas y être vraie. Un modèle est toujours une structure cohérente et complète : chaque formule close y est soit vraie soit fausse, et aucune formule ne peut y être vraie en même temps que sa négation. On peut utiliser cela pour montrer le théorème de cohérence.

**Théorème 2.20**
*Soit $T$ une théorie, et soit $F(x_1, \ldots, x_n)$ une formule dans le langage de cette théorie. Si $T \vdash F(x_1, \ldots, x_n)$, alors tout modèle de $T$ est aussi un modèle de $\forall x_1 \ldots \forall x_n \; F(x_1, \ldots, x_n)$.*

Preuve. On procède sans peine par induction sur la taille d'une démonstration. Supposons que ce soit le cas pour toute démonstration $G_1, \ldots, G_n$ dans $T$. Soit $G_1, \ldots, G_{n+1}$ une démonstration dans $T$. Par hypothèse d'induction, tout modèle de $T$ est modèle de chaque formule $\forall \overline{x} G_i$ pour $i < n+1$, où la notation $\forall \overline{x} G_i$ signifie que l'on quantifie universellement sur chaque variable libre de $G_i$ (quand il y en a). Soit $\mathcal{M}$ un modèle de $T$. Si $G_{n+1}$ est un axiome de $T$, alors c'est une formule close, et $\forall \overline{x} G_{n+1}$ est bien évidemment vraie dans $\mathcal{M}$. Si $G_{n+1}$ est un axiome de l'égalité (de type (e) (f) ou (g), ci-dessus), alors $\forall \overline{x} G_{n+1}$ est vraie dans $\mathcal{M}$ par le fait que l'égalité de $\mathcal{M}$ est toujours la vraie égalité.

Si $G_{n+1}$ est une tautologie de la logique propositionnelle ou un axiome de type (a)(b)(c) ou (d) du calcul des prédicats, alors $\forall \overline{x} G_{n+1}$ est vraie par définition de la satisfaction dans un modèle (les détails sont laissés au lecteur ; notons que pour (d) on utilise le fait que le modèle soit non vide).

Si $G_{n+1}$ est obtenue par généralisation sur $G_i$ pour $i < n+1$ — en particulier, $G_{n+1}$ est de la forme $\forall y G_i$ —, alors $\mathcal{M}$, vérifiant $\forall \overline{x} G_i$, vérifie également $\forall \overline{x} G_{n+1}$ (qui est en fait la même formule). Enfin, si $G_{n+1}$ est obtenu par Modus Ponens via $G_i \to G_{n+1}$ et $G_i$ pour $i < n+1$, alors $\mathcal{M}$ vérifie $\forall \overline{x} G_i$ et $\forall \overline{x}(G_i \to G_{n+1})$. Par définition de la satisfaction, $\mathcal{M}$ vérifie donc $\forall \overline{x} G_{n+1}$. ∎

Le théorème précédent peut se résumer de la manière suivante : « on ne peut démontrer que des choses vraies ». C'est une bonne nouvelle, de laquelle on peut déduire notre théorème de cohérence.

2. Aphorisme cher au professeur René Cori, qui enseigna aux auteurs de ce livre les principes du présent chapitre.

**Corollaire 2.21 (Théorème de cohérence)**
*Si un système axiomatique admet un modèle, alors il est cohérent.*

PREUVE. On le montre par contraposée. Si $T \vdash F \wedge \neg F$ pour une formule close $F$, alors tout modèle de $T$ est modèle de $F \wedge \neg F$. Comme il n'existe aucun modèle de $F \wedge \neg F$, alors $T$ n'a pas de modèle. ∎

On vérifie sans peine l'existence d'un modèle des axiomes de la logique : l'ensemble $\{1\}$ avec la relation d'égalité $1 = 1$. Cela montre que les axiomes de la logique sont cohérents et ne peuvent démonter $\perp$.

On vérifie aussi sans peine l'existence d'un modèle pour l'arithmétique de Peano, à savoir $\mathbb{N}$ muni des fonctions usuelles d'addition et de multiplication, ainsi que des relations usuelles $<$ et $=$ sur les entiers. Voilà une deuxième bonne nouvelle, les axiomes de l'arithmétique de Peano sont eux aussi cohérents. Nous examinerons le sens de cette affirmation un peu plus loin, notamment à la lumière du deuxième théorème d'incomplétude de Gödel et de ses implications.

### 2.5. Modèles et théorème de complétude

Si le système de déduction à la Hilbert que nous avons donné — avec ses axiomes de la logique et ses règles de déduction — est bien cohérent, comment savoir en revanche qu'il est suffisamment puissant ? Après tout, les deux règles d'inférence semblent former un outil de travail bien maigre. Peut-on réellement tout démontrer avec ce système ? Nous allons voir que c'est le cas, via un théorème démontré, par Gödel dans sa thèse de doctorat, qui peut être vu comme la réciproque du théorème de cohérence : tout ce qui est universellement vrai est démontrable.

**Théorème 2.22 (Théorème de complétude de Gödel)**
*Soit $T$ une théorie dans un langage dénombrable. Si $T$ est cohérente, alors $T$ a un modèle.*

Notons que le théorème de complétude pour des langages indénombrables peut aussi se démontrer en utilisant l'axiome du choix. Nous n'en montrons que la version dénombrable, qui nous suffira amplement. Le théorème de complétude de Gödel est plus difficile à montrer que le théorème de cohérence, lequel est une simple vérification de routine. Il s'agit ici de construire un modèle d'une théorie $T$, à partir du simple fait que $T \nvdash \perp$. L'idée de la preuve passe par la création d'une théorie dite *complète*.

**Définition 2.23.** Une théorie $T$ est dite *complète* si $T \vdash F$ ou $T \vdash \neg F$ pour toute formule close $F$ dans le langage de $T$. ♢

**Proposition 2.24.** Soit $\mathcal{L}$ un langage dénombrable. Toute théorie de $\mathcal{L}$ cohérente peut être étendue en une théorie complète et cohérente. ⋆

La preuve de la proposition ci-dessus utilise le lemme 2.10 que nous reformulons ici avec la notation $\vdash$ introduite depuis.

**Lemme (Lemme 2.10 de déduction).** Soit $T \cup \{F\}$ une théorie et $G$ une formule. Supposons $T \cup \{F\} \vdash G$. Alors, $T \vdash F \to G$. ⋆

Preuve de la proposition 2.24. Étant donné une théorie $T$ cohérente dans un langage dénombrable, on en construit inductivement une extension $T'$ complète et cohérente. Soit $T_0 = T$. À l'étape $n$, supposons qu'une théorie cohérente $T_n$ soit définie. Soit $F_n$ la $n$-ième formule close de notre langage. Si $T_n \vdash F_n$, on définit $T_{n+1} = T_n \cup \{F_n\}$. Si $T_n \vdash \neg F_n$, on définit $T_{n+1} = T_n \cup \{\neg F_n\}$. Dans ces deux premiers cas, la cohérence de $T_{n+1}$ découle de la cohérence de $T_n$ et du lemme de déduction.

Si jamais $T_n$ ne prouve ni $F_n$ ni $\neg F_n$, alors on définit $T_{n+1} = T_n \cup \{F_n\}$. Supposons par l'absurde que $T_n \cup \{F_n\} \vdash \bot$. Alors, d'après le lemme de déduction, $T_n \vdash F_n \to \bot$, et donc $T_n \vdash \neg F_n$ (par contraposée et Modus Ponens), ce qui contredit le fait que $T$ ne prouve pas $\neg F_n$. Notons que l'on pourrait tout aussi bien définir $T_{n+1} = T_n \cup \{\neg F_n\}$.

On définit enfin $T' = \bigcup_n T_n$. La cohérence de $T'$ vient alors du fait qu'une démonstration dans une théorie n'utilise qu'un nombre fini de ses axiomes : si $T' \vdash \bot$, alors il existe forcément $n$ tel que $T_n \vdash \bot$, Comme chaque théorie $T_n$ est cohérente, alors $T'$ doit être cohérente. ∎

Preuve du théorème de complétude. La preuve que nous donnons est due à Léon Henkin [85], et repose sur la création d'une théorie complète, cohérente, et possédant ce que l'on appelle des *témoins de Henkin*. Nous allons d'abord montrer l'affirmation suivante.

« Soit $T$ une théorie dans un langage $\mathcal{L}$, et soit $c$ un symbole de constante qui n'apparaît pas dans $\mathcal{L}$. Supposons $T \cup \{\exists x F(x) \to F(c)\} \vdash \bot$. On a alors $T \vdash \bot$ . »

Comme $T \cup \{\exists x F(x) \to F(c)\} \vdash \bot$, on a alors, par le lemme de déduction, contraposée et Modus Ponens, $T \vdash \exists x F(x) \wedge \neg F(c)$. En particulier, $T \vdash \neg F(c)$. Soit à présent une variable $z$ qui n'intervient pas dans la preuve de $\neg F(c)$. Comme la constante $c$ n'apparaît pas dans la théorie $T$, elle ne peut être introduite dans la démonstration que par un axiome de la logique. Chacun de ces axiomes reste valide en remplaçant $c$ par $z$. On

laisse au lecteur le soin de vérifier que si l'on remplace $c$ par $z$ pour chaque étape de la démonstration, on obtient une démonstration valide de $\neg F(z)$. Par la règle de généralisation, on obtient finalement $T \vdash \forall x\, \neg F(x)$. Comme également $T \vdash \exists x F(x)$, alors $T \vdash \bot$.

Passons à présent à la preuve du théorème de complétude. Soit $T$ une théorie dans un langage dénombrable $\mathcal{L}$. Montrons comment construire un modèle. On définit $T_0 = T$ et $\mathcal{L}_0 = \mathcal{L}$. À l'étape $n \in \mathbb{N}$, supposons que l'on a défini une théorie cohérente $T_{2n}$ dans un langage $\mathcal{L}_{2n}$. Soit $\mathcal{L}_{2n+1}$ le langage $\mathcal{L}_{2n}$ auquel on ajoute un nouveau symbole de constante $c_G$ pour toute formule $G$ de $T_{2n}$ de la forme $\exists x F(x)$. Soit $T_{2n+1}$ la théorie $T_{2n}$ à laquelle on ajoute les énoncés $\exists x F(x) \to F(c_G)$ pour tout énoncé $G$ de $T_{2n}$ de la forme $\exists x F(x)$. Notons que par l'affirmation ci-dessus, comme $T_{2n}$ est cohérente, à chaque ajout d'axiome de la forme $\exists x F(x) \to F(c_G)$, la théorie reste cohérente. Donc, $T_{2n+1}$ est cohérente. Finalement, soit $\mathcal{L}_{2n+2}$ égal à $\mathcal{L}_{2n+1}$, et en utilisant la proposition 2.24, soit $T_{2n+2}$ la complétion de $T_{2n+1}$ pour le langage $\mathcal{L}_{2n+2}$. Soient $\mathcal{T}_\omega = \bigcup_n T_n$ et $\mathcal{L}_\omega = \bigcup_n \mathcal{L}_n$. Notons que $\mathcal{T}_\omega$ est une théorie complète et cohérente dans le langage $\mathcal{L}_\omega$, qui contient de plus un énoncé de la forme $\exists x F(x) \to F(c)$ pour chacun des énoncés de la forme $\exists x F(x)$ de $T_\omega$, où $c$ est un symbole de constante de $\mathcal{L}_\omega$. Les fameux *témoins de Henkin* sont les nouveaux symboles de constantes ainsi introduits.

L'ensemble sous-jacent $M$ de notre modèle $\mathcal{M}$ est l'ensemble des termes clos de $\mathcal{L}_\omega$, quotienté par la relation d'égalité. Formellement, si $(t_n)_{n\in\mathbb{N}}$ est la liste des termes clos de $\mathcal{L}_\omega$, alors $M = \{t_n \ : \ \forall i < n\ (\neg t_i = t_n) \in T_\omega\}$.

Notons que les symboles de fonctions de $\mathcal{L}$ ont une interprétation claire dans $M$. Par exemple, si $f$ est un symbole de fonction unaire de $\mathcal{L}$, alors sa fonction correspondante $f^{\mathcal{M}} : M \to M$ est définie par $f^{\mathcal{M}}(t) = q$ pour $q$ l'élément de $M$ qui est égal au terme clos $f(t) \in \mathcal{L}_\omega$, c'est-à-dire tel que $(f(t) = q) \in T_\omega$.

La théorie $T_\omega$ étant complète et cohérente, pour tout symbole de relation $m$-aire $R$ de $\mathcal{L}$ et tout élément $t_1, \ldots, t_m \in M$, exactement un des énoncés parmi $R(t_1, \ldots, t_m)$ ou $\neg R(t_1, \ldots, t_m)$ est dans $T_\omega$. Cela induit une interprétation $R^{\mathcal{M}}$ du symbole de relation $R$ de $\mathcal{L}$ dans $\mathcal{M}$.

Il reste à montrer par induction sur les formules que tous les énoncés de $T_\omega$ sont satisfaits dans $\mathcal{M}$ (et donc aussi ceux de $T$). Sans perte de généralité, on ne traite que les formules sous forme prénexe et où le symbole de négation n'apparaît que devant les formules atomiques. Par définition des relations dans $\mathcal{M}$, c'est bien le cas pour les formules atomiques et leurs négations. Si $F_1 \wedge F_2 \in T_\omega$, alors comme $T_\omega$ est complète $F_1, F_2 \in T_\omega$. Par hypothèse d'induction, $\mathcal{M} \models F_1$ et $\mathcal{M} \models F_2$, et donc $\mathcal{M} \models F_1 \wedge F_2$.

Si $F_1 \vee F_2 \in T_\omega$, alors comme $T$ est complète $F_1 \in T_\omega$ ou $F_2 \in T_\omega$ (sinon, par complétude, $\neg F_1, \neg F_2 \in T_\omega$, ce qui contredit $F_1 \vee F_2$). Par hypothèse d'induction, $\mathcal{M} \models F_1 \vee F_2$. Si $\forall x F(x) \in T_\omega$, alors comme $T_\omega$ est complète $F(t)$ est dans $T_\omega$ pour tout terme clos $t$ de $\mathcal{L}_\omega$, et donc tout élément de $M$. Par hypothèse d'induction, $\mathcal{M} \models F(t)$ pour tout $t \in M$, et donc $\mathcal{M} \models \forall x\ F(x)$. Si $\exists x F(x) \in T_\omega$, alors $\exists x F(x) \to F(c) \in T_\omega$ pour un symbole de constante $c \in \mathcal{L}_\omega$. En particulier, $F(c) \in T_\omega$ par Modus Ponens. Soit $t \in M$ un terme clos de $\mathcal{L}_\omega$ tel que $(t = c) \in T_\omega$. Par les axiomes de l'égalité, on a $F(t)$. Par hypothèse d'induction, $\mathcal{M} \models F(t)$. Donc, $\mathcal{M} \models \exists x F(x)$. ∎

Le théorème de complétude est souvent utilisé sous la forme du corollaire suivant.

**Corollaire 2.26**
*Si tout modèle d'une théorie $T$ est modèle d'une formule $F$, alors $T \vdash F$.*

Preuve. Si l'on a $T \cup \{\neg F\} \vdash \bot$, alors $T \vdash \neg F \to \bot$ par le lemme de déduction, et donc $T \vdash F$.

Supposons à présent $T \nvdash F$ ; alors, par la ligne ci-dessus $T \cup \{\neg F\} \nvdash \bot$. D'après le théorème de complétude, il existe donc un modèle de $T \cup \{\neg F\}$, c'est-à-dire un modèle de $T$ qui ne soit pas modèle de $F$. ∎

Le théorème de complétude implique en particulier que l'on ne peut pas faire mieux que le système à la Hilbert que nous avons présenté : supposons que dans ce système les axiomes seuls de la logique ne suffisent pas à démontrer une formule $F$, autrement dit $\nvdash F$ (à partir d'une théorie vide). Supposons qu'un système de démonstration plus puissant et cohérent existe tel que $\vdash^* F$, où $\vdash^*$ est la notion de preuve dans ce système. Alors également, $\vdash^* \neg F \to \bot$, et donc $\neg F \vdash^* \bot$. À présent, comme $\nvdash F$, il existe d'après le théorème de complétude un modèle de $\neg F$, et notre modèle est donc d'après le théorème de cohérence pour $\vdash^*$ un modèle de $\bot$, ce qui est impossible.

## 3. Théorèmes d'incomplétudes de Gödel

Nous rentrons à présent dans le vif du sujet, via les théorèmes d'incomplétudes de Gödel, qui reposent entre autres choses sur un codage des ensembles calculatoirement énumérables par des formules de l'arithmétique.

### 3.1. Formules de l'arithmétique de Peano

Nous avons défini dans le chapitre 5 une hiérarchie de complexité sur les ensembles, appelée *hiérarchie arithmétique*. Nous allons maintenant donner tout son sens à cette appellation en définissant une hiérarchie syntaxique des formules de l'arithmétique, qui coïncide avec la hiérarchie arithmétique, dans le sens où un ensemble $A$ est $\Sigma_n^0$ ssi il est définissable par une formule $\Sigma_n$ de l'arithmétique de Peano.

**Définition 3.1.** Une formule de l'arithmétique de Peano est $\Delta_0$ si les quantifications qu'elle comporte sont toutes bornées, c'est-à-dire de la forme $\exists x < t$ et $\forall x < t$, avec $t$ un terme où la variable $x$ n'apparaît pas librement. Notons que les formules $\exists x < t\ F(x)$ et $\forall x < t\ F(x)$ se traduisent respectivement par $\exists x(x < t \wedge F(x))$ et $\forall x(x < t \to F(x))$. ♢

Étant donné $F(x_1, \ldots, x_n)$ une formule $\Delta_0$ de l'arithmétique de Peano, la restriction sur les quantifications fait de l'ensemble

$$\{(x_1, \ldots, x_n) \in \mathbb{N} :\ F(x_1, \ldots, x_n)\}$$

un ensemble calculable. Cela découle par exemple directement de la clôture des prédicats primitifs récursifs par conjonction, disjonction et quantification bornée (voir l'exemple 6-3.16 et l'exercice 6-3.19). On définit une hiérarchie de complexité sur les formules de l'arithmétique de Peano analogue à la hiérarchie arithmétique de la définition 5-1.1.

**Définition 3.2**

1. Une formule $F(x_1, \ldots, x_m)$ de l'arithmétique de Peano est $\Sigma_n$ si
$$F(x_1, \ldots, x_m) = \overbrace{\exists y_1 \forall y_2 \ldots Q y_n}^{n \text{ quantificateurs}}\ G(x_1, \ldots, x_m, y_1, \ldots, y_n)$$
pour $G(x_1, \ldots, x_m, y_1, \ldots, y_n)$ une formule $\Delta_0$, où $Q$ vaut $\exists$ si $n$ est impair, et $\forall$ si $n$ est pair.
2. Une formule $F(x_1, \ldots, x_m)$ de l'arithmétique de Peano est $\Pi_n$ si
$$F(x_1, \ldots, x_m) = \overbrace{\forall y_1 \exists y_2 \ldots Q y_n}^{n \text{ quantificateurs}}\ G(x_1, \ldots, x_m, y_1, \ldots, y_n)$$
pour $G(x_1, \ldots, x_m, y_1, \ldots, y_n)$ une formule $\Delta_0$, où $Q$ vaut $\forall$ si $n$ est impair, et $\exists$ si $n$ est pair. ♢

Nous verrons que les ensembles définissables par une formule $\Sigma_n$ (resp. $\Pi_n$) de l'arithmétique de Peano coïncident avec les ensembles $\Sigma_n^0$ (resp. $\Pi_n^0$) de la définition 5-1.1. Nous commençons pour cela par montrer quelques propriétés de clôture analogues à celles des propositions 5-1.6 à 5-1.9.

**Proposition 3.3.** Soient $F(\overline{a},x)$, $F_1(\overline{a},x)$ et $F_2(\overline{a},x)$ des formules $\Sigma_n$ (resp $\Pi_n$). Alors, chacune des formules suivantes est prouvablement équivalente (en usant des axiomes de l'arithmétique) à une formule $\Sigma_n$ (resp. $\Pi_n$) :

- ▷ $F_1(\overline{a},x) \wedge F_2(\overline{a},x)$, $F_1(\overline{a},x) \vee F_2(\overline{a},x)$ ;
- ▷ $\exists x < b\ F(\overline{a},x)$, $\forall x < b\ F(\overline{a},x)$ ;
- ▷ $\exists x\ F(\overline{a},x)$ (resp. $\forall x\ F(\overline{a},x)$). ⋆

Preuve. Soient les équivalences suivantes :

$$F(\overline{a},x) \equiv \exists y G(\overline{a},x,y),\ F_1(\overline{a},x) \equiv \exists y G_1(\overline{a},x,y) \text{ et } F_2(\overline{a},x) \equiv \exists y G_2(\overline{a},x,y).$$

Il s'en déduit alors les nouvelles équivalences suivantes :

$$\begin{array}{rcl} F_1(\overline{a},x) \wedge F_2(\overline{a},x) & \leftrightarrow & \exists y\ \exists y_1, y_2 < y\ (G_1(\overline{a},x,y_1) \wedge G_2(\overline{a},x,y_2)) \\ F_1(\overline{a},x) \vee F_2(\overline{a},x) & \leftrightarrow & \exists y\ (G_1(\overline{a},x,y) \vee G_2(\overline{a},x,y)) \\ \exists x < b\ F(\overline{a},x) & \leftrightarrow & \exists y\ \exists x < b\ G(\overline{a},x,y) \\ \forall x < b\ F(\overline{a},x) & \leftrightarrow & \exists z\ \forall x < b\ \exists y < z\ G(\overline{a},x,y) \\ \exists x\ F(\overline{a},x) & \leftrightarrow & \exists z\ \exists x < z\ \exists y < z\ G(\overline{a},x,y). \end{array}$$

À présent, si $F, F_1, F_2$ sont $\Sigma_1$ avec $G, G_1, G_2$ $\Delta_0$, les équivalences ci-dessus montrent la proposition pour le cas $\Sigma_1$. Par passage à la négation, les équivalences tiennent aussi pour le cas $\Pi_1$. Supposons la proposition vraie pour les cas $\Sigma_n$ et $\Pi_n$. Alors, les équivalences ci-dessus pour $F, F_1, F_2$ des formules $\Sigma_{n+1}$, avec $G, G_1, G_2$ $\Pi_n$, impliquent —en utilisant les hypothèses d'induction sur $G, G_1, G_2$— la proposition pour le cas $\Sigma_{n+1}$. Par négation, la proposition est vraie pour le cas $\Pi_{n+1}$. ∎

Notons que la quatrième équivalence dans la preuve ci-dessus, en l'occurrence l'équivalence $\forall x < b\ F(\overline{a},x) \leftrightarrow \exists z\ \forall x < b\ \exists y < z\ G(\overline{a},x,y)$, est la moins triviale de toutes : les quatre autres utilisent simplement le fait que deux entiers ont toujours un majorant, alors que celle-ci requiert l'utilisation de l'induction. C'est quelque chose que nous étudierons en détail dans la section 23-3.

Passons à présent à l'équivalence annoncée. Étant donné $\exists y\ F(x_1,\ldots,x_n,y)$ une formule $\Sigma_1$ de l'arithmétique de Peano avec $F$ qui est $\Delta_0$, l'ensemble

$$\{(x_1,\ldots,x_n) \in \mathbb{N} :\ \exists y\ F(x_1,\ldots,x_n,y)\}$$

est un ensemble calculatoirement énumérable : on teste petit à petit la formule $F$ sur tous les $(n+1)$-uplets $x_1,\ldots,x_n,y$ et, quand on en trouve un pour lequel $F$ est vraie, on énumère $(x_1,\ldots,x_n)$. Gödel a montré que tout ensemble calculatoirement énumérable pouvait en fait être représenté sous cette forme.

**Théorème 3.4 (Gödel)**
*Un ensemble d'entiers $A \subseteq \mathbb{N}$ est c. e. si, et seulement si, il existe $F(n)$ une formule $\Sigma_1$ de $\mathcal{L}_{\mathrm{PA}}$ telle que $n \in A$ ssi $\mathbb{N} \vDash F(n)$.*

Nous allons montrer le théorème 3.4 en nous basant sur le modèle des fonctions générales récursives, qui coïncident, comme nous l'avons vu dans le chapitre 6, avec les fonctions calculables. Nous allons montrer que toute fonction partielle générale récursive $f : \mathbb{N}^k \to \mathbb{N}$ est *représentée* par une formule $\Sigma_1$ de l'arithmétique $F(n_1, \ldots, n_k)$, c'est-à-dire

$$\{(\overline{n}, r) \in \mathbb{N}^{k+1} : f(\overline{n}) \downarrow = r\} \;=\; \{(\overline{n}, r) \in \mathbb{N}^{k+1} : \mathbb{N} \vDash F(\overline{n}, r)\}.$$

Comme tout ensemble c. e. est le domaine d'une fonction partielle, cela démontre le théorème. La difficulté principale se trouve dans la gestion du schéma de récursion primitive, pour lequel nous avons besoin de coder des listes d'entiers par des formules de l'arithmétique. Gödel a recours pour cela à une utilisation astucieuse d'un résultat d'arithmétique modulaire : le théorème des restes chinois.

**Lemme 3.5 (Théorème des restes chinois).** Soit une suite quelconque d'entiers $(a_0, \ldots, a_n)$, et soit $(p_0, \ldots, p_n)$ une suite d'entiers deux à deux premiers entre eux avec $p_i \geqslant a_i$ pour $i \leqslant n$. Alors, il existe un entier $b$ tel que $a_i$ est le reste de la division euclidienne de $b$ par $p_i$ pour tout $i$. ⋆

Le théorème des restes chinois va permettre à Gödel de coder des listes d'entiers de taille arbitraire. Notons que ce n'est pas la seule manière d'établir un système de codage/décodage des listes par des formules de l'arithmétique, et nous en verrons une autre dans la section 23-4.

**Lemme 3.6 (Fonction $\beta$ de Gödel).** Il existe une fonction

$$\beta : \mathbb{N}^3 \to \mathbb{N}$$

représentée par une formule $\Delta_0$ telle que pour tout $n$ et toute suite d'entiers $(a_0, \ldots, a_n)$ il existe des entiers $a, b \in \mathbb{N}$ pour lesquels $\beta(a, b, i) = a_i$ pour tout $i \leqslant n$. ⋆

Preuve. La formule $B(a, b, i, r)$ qui représente $\beta$ est la suivante : « $r$ est le reste de la division euclidienne de $b$ par $a(i+1)+1$ ». La formule est bien $\Delta_0$ : $r < a \times (i+1) + 1 \;\wedge\; \exists c < b \; c \times (a \times (i+1) + 1) + r = b$. Soit une suite d'entiers $(a_0, \ldots, a_n)$. Montrons l'existence d'entiers $a, b$ tels que cette formule définit bien la fonction $(a, b, i) \mapsto a_i$.

Soit $m$ tel que $m > \max\{a_i \;:\; i \leqslant n\}$ et $m > n$. Posons $a = m!$. Nous allons utiliser le théorème des restes chinois avec $p_i = a(i+1)+1$. Montrons que ces nombres sont deux à deux premiers.

Supposons par l'absurde qu'un nombre premier $p$ divise $p_i$ et $p_j$ avec $i < j$. Alors aussi, $p$ divise $p_j - p_i = a(j-i)$. Comme $p$ est premier, alors $p$ divise $a$ ou $p$ divise $j - i$. Comme $a = m!$ avec $m > n \geqslant j > i$ alors $j - i$ divise $a$ et donc dans tous les cas $p$ divise $a$. Donc, $p$ divise $a(i+1)$. Comme $p$ divise également $a(i+1)+1$, alors, $p$ divise $a(i+1)+1-a(i+1) = 1$, ce qui est une contradiction. Les nombres $p_i$ sont donc premiers entre eux.

On a bien $p_i = a(i+1)+1 > m > a_i$ pour tout $i$. D'après le théorème des restes chinois, il existe un entier $b$ tel que pour tout $i$ l'entier $a_i$ est le reste de la division euclidienne de $b$ par $a(i+1)+1$. ∎

PREUVE DU THÉORÈME 3.4. On montre que toute fonction générale récursive partielle est représentée par une formule $\Sigma_1$ de l'arithmétique. On laisse au lecteur le soin de montrer que c'est bien le cas pour les fonctions de base (projections, fonctions constantes et fonction successeur). On utilise sans le mentionner pour chacun des trois schémas à venir la proposition 3.3 afin d'obtenir une formule $\Sigma_1$ qui représente notre fonction.

*Schéma de composition.* Soit

$$f(\overline{x}) = g(h_1(\overline{x}), \ldots, h_k(\overline{x}))$$

pour des fonctions $g, h_1, \ldots, h_k$ représentées par des formules $G, H_1, \ldots, H_k$. Alors, $f$ est représentée par la formule

$$F(\overline{x}, r) \equiv \exists y_1, \ldots, y_k \; H_1(\overline{x}, y_1) \wedge \cdots \wedge H_k(\overline{x}, y_k) \wedge G(y_1, \ldots, y_k, r).$$

*Schéma de minimisation.* Soit

$$f(\overline{x}) = \min\{a \in \mathbb{N} \; : \; \forall i \leqslant a \; g(\overline{x}, i) \downarrow \; \wedge \; g(\overline{x}, a) = 0\}$$

pour $g$ représentée par une formule $G$. Alors, $f$ est représentée par la formule :

$$F(\overline{x}, a) \equiv G(\overline{x}, a, 0) \wedge \forall i < a \; \exists r \neq 0 \; G(\overline{x}, i, r).$$

*Schéma de récursion primitive.* C'est ici que nous aurons besoin de la fonction $\beta$ de Gödel, représentée par la formule $B$. Soit $f$ défini par :

$$\begin{aligned} f(\overline{x}, 0) &= g(\overline{x}) \\ f(\overline{x}, n+1) &= h(\overline{x}, n, f(\overline{x}, n)) \end{aligned}$$

pour des fonctions $g, h$ représentées par des formules $G, H$. Alors, $f$ est représentée par la formule $F(\overline{x}, n, r)$ suivante :

$$\begin{aligned} \exists a, b \quad & B(a, b, n, r) \wedge \exists a_0 \; \big(B(a, b, 0, a_0) \wedge G(\overline{x}, a_0)\big) \wedge \forall i < n \\ & \exists a_i, a_{i+1} \; \big(B(a, b, i, a_i) \wedge B(a, b, i+1, a_{i+1}) \wedge H(\overline{x}, i, a_i, a_{i+1})\big). \end{aligned}$$

Afin de voir que $f$ est bien représentée par $F$, il nous faut remarquer que $B$ tel que définie dans le lemme précédent est toujours une formule fonctionnelle.

En effet, quelles que soient les valeurs de $a, b, i$, il y a toujours au plus un élément $r$ tel que $B(a,b,i,r)$ est vrai. Ainsi, si la formule $F(\overline{x},n,r)$ est vérifiée, il existe bien une suite $a_0,\ldots,a_n$ telle que $g(\overline{x})=a_0$ et $h(\overline{x},i,a_i)=a_{i+1}$, avec $a_n = r$, le résultat de $f(\overline{x}, n)$. D'après le lemme précédent, il existe bien pour tout $n$ des entiers $a, b$ qui codent via la fonction $\beta$ la suite de valeurs $(f(\overline{x}, 0), f(\overline{x}, 1), \ldots, f(\overline{x}, n))$.
La formule $F$ représente donc bien la fonction $f$. ∎

On montre facilement à partir du théorème 3.4 qu'un ensemble d'entiers est $\Sigma_n^0$ (resp. $\Pi_n^0$) si, et seulement si, il est décrit par une formule $\Sigma_n$ (resp. $\Pi_n$) de l'arithmétique.

**Codage des suites finies**

Il existe de nombreuses manières de coder des suites finies d'entiers à l'aide d'entiers naturels. La plupart de ces techniques ont recours à des fonctions primitives récursives, ce qui demande de montrer au préalable qu'elles sont représentables par des formules simples de l'arithmétique. Le théorème des restes chinois permet de réaliser un codage simple des suites finies à base de la division euclidienne, qui s'exprime par un prédicat $\Delta_0$ immédiat (voir le lemme 3.6).

### 3.2. Démonstrations et calcul

Une démonstration dans notre système de déduction à la Hilbert repose sur un système de règles bien précis, et il est aisé de créer un programme informatique qui prend en paramètre une démonstration — via un codage approprié — et qui vérifie en un temps fini si la démonstration est valide ou non. En effet, les règles d'inférence sont claires, quant aux axiomes de la logique, nous avons les tautologies du calcul propositionnel, quatre schémas d'axiomes pour le calcul des prédicats, et trois schémas d'axiomes pour l'égalité. Il est aisé de vérifier si une phrase du calcul propositionnel — par exemple, de la forme $(F \to G) \leftrightarrow (\neg G \to \neg F)$ — est une tautologie, en s'assurant que la phrase est toujours vraie pour n'importe quelle valeur de vérité pour $F$ et $G$. Il est également facile de vérifier si une phrase correspond à l'un des schémas d'axiomes de l'égalité ou du calcul des prédicats. On en déduit le théorème suivant.

**Théorème 3.7 (Gödel)**
*Étant donné une théorie calculatoirement énumérable $T$ dans le langage $\mathcal{L}_{PA}$, l'ensemble des formules $F$ telles que $T \vdash F$ est calculatoirement énumérable.*

Il suffit en effet de faire une recherche sur toutes les démonstrations possibles à partir des axiomes de $T$ et de lister toutes les formules qu'elles démontrent.

L'objectif de Hilbert était de montrer que toute vérité arithmétique était prouvable, l'arithmétique de Peano étant supposément un système suffisant pour le faire. Si tel était le cas, alors on pourrait créer un algorithme permettant de décider si un énoncé mathématique $F$ est prouvable ou réfutable : il suffirait de lister tous les énoncés prouvés par l'arithmétique de Peano jusqu'à trouver $F$ ou $\neg F$.

Gödel a montré que ce n'était pas possible, ni pour l'arithmétique de Peano, ni pour aucune théorie calculatoirement énumérable et cohérente contenant l'arithmétique de Peano (il fallut toutefois l'aide de Rosser pour cette dernière étape). Nous commençons par prouver un lemme qui nous aidera dans la suite.

**Lemme 3.8.** Si $\mathbb{N} \vDash F$ où $F$ est une formule close $\Sigma_1$, alors $\mathrm{PA} \vdash F$. ⋆

Preuve. La formule $F$ est de la forme $\exists x_1 \dots \exists x_n \; G(x_1, \dots, x_n)$ pour $G$ une formule $\Delta_0$. Si $F$ est vraie dans $\mathbb{N}$, alors il existe des entiers $a_1, \dots, a_n \in \mathbb{N}$ tels que $G(a_1, \dots, a_n)$ est vrai dans $\mathbb{N}$. On montre facilement par induction que si une formule $\Delta_0$ et paramétrée dans $\mathbb{N}$ est vraie dans $\mathbb{N}$, alors elle est démontrable dans PA. À titre informel, pour une quantification existentielle bornée, il suffit de prendre un entier témoin pour cette quantification et de montrer la formule avec ce témoin ; et, pour une quantification universelle bornée par un entier $a$, il suffit de montrer que la formule est vraie pour chaque entier inférieur à $a$. ∎

Nous avons à présent les ingrédients nécessaires pour montrer le premier théorème d'incomplétude.

**Théorème 3.9 (Premier théorème d'incomplétude de Gödel)**
*Soit $T \supseteq \mathrm{PA}$ une théorie c. e. cohérente, telle que si $T$ démontre une formule $\Sigma_1$, alors $\mathbb{N}$ est modèle de cette formule. Alors, il existe $F$ une formule $\Sigma_1$ telle que $T \nvdash F$ et $T \nvdash \neg F$.*

Preuve. Soit $(\Phi_e)_{e \in \mathbb{N}}$ une énumération des fonctions calculables. D'après le théorème 3.4, il existe $F(e)$ une formule $\Sigma_1$ de $\mathcal{L}_{\mathrm{PA}}$ telle que

$$\{e \in \mathbb{N} \; : \; \mathbb{N} \vDash F(e)\} = \{e \in \mathbb{N} \; : \; \exists t \; \Phi_e(e)[t] \downarrow\}.$$

Supposons que pour tout $e$ on ait $T \vdash F(e)$ ou $T \vdash \neg F(e)$. Notons que si $\exists t \; \Phi_e(e)[t] \downarrow$, alors $\mathbb{N} \vDash F(e)$, et donc $\mathrm{PA} \vdash F(e)$ d'après le lemme 3.8. Comme $\mathrm{PA} \subseteq T$, on a aussi $T \vdash F(e)$.

À présent, si $\forall t \; \Phi_e(e)[t] \uparrow$, alors on a $\mathbb{N} \vDash \neg F(e)$. D'après notre hypothèse, on ne peut avoir $T \vdash F(e)$, car on aurait alors $\mathbb{N} \vDash F(e)$ ce qui contredit $\mathbb{N} \vDash \neg F(e)$. Comme $T$ est complète, on a donc $T \vdash \neg F(e)$.

Il s'ensuit que l'ensemble c. e. $\{e \in \mathbb{N} \ : \ T \vdash \neg F(e)\}$ coïncide avec l'ensemble $\{e \in \mathbb{N} \ : \ \forall t \ \Phi_e(e)[t] \uparrow\}$, ce qui fait du complémentaire de l'arrêt un ensemble c. e. On a là une contradiction. ∎

Nous remarquons plusieurs choses. En premier lieu, nous avons été obligés de nous restreindre aux théories $T$ dites 1-cohérentes, c'est-à-dire ne démontrant pas de formules $\Sigma_1$ fausses dans $\mathbb{N}$. Nous verrons bientôt qu'il existe bien d'autres modèles possibles que $\mathbb{N}$ pour PA, et autant de théories non 1-cohérentes que l'on souhaite. Il s'agit donc d'une restriction bien ennuyeuse.

Ensuite, la preuve montre en substance qu'une théorie complète et 1-cohérente permet de calculer $\emptyset'$. Nous avons déjà vu que n'importe quel degré PA permet de calculer une extension complète et cohérente de PA, et comme il y a des degrés PA qui ne calculent pas $\emptyset'$, il n'y a en fait aucun espoir de montrer le théorème de Gödel en réduisant le problème de l'arrêt à toute théorie complète et cohérente qui étend PA. L'astuce pour se sortir de cette situation fut trouvée par Rosser, l'idée étant en substance d'utiliser le fait que si une théorie démontre $\exists t \ \Phi_e(e)[t] \downarrow = 0$ (que ce soit vrai dans $\mathbb{N}$ ou pas), alors elle ne peut pas montrer dans le même temps $\exists t \ \Phi_e(e)[t] \downarrow = 1$, à supposer bien entendu que les formules $\Sigma_1$ permettant de parler des fonctions calculables intègrent bien le fait qu'une fonction a au plus une valeur sur son entrée, ce qui est bien le cas en pratique.

**Théorème 3.10 (Théorème d'incomplétude de Gödel-Rosser)** *Soit $T \supseteq$ PA une théorie c. e. et cohérente. Alors, il existe $F$ une formule $\Sigma_1$ telle que $T \nvdash F$ et $T \nvdash \neg F$.*

Preuve. Soit $(\Phi_e)_{e\in\mathbb{N}}$ une énumération des fonctions calculables. Supposons par l'absurde que l'on ait $T \vdash F$ ou $T \vdash \neg F$ pour toute formule $F$ $\Sigma_1$. Nous allons alors calculer une fonction totale $f : \mathbb{N} \to \{0, 1\}$ qui est $\mathrm{DNC}_2$, c'est-à-dire telle que pour tout entier $e$ on ait $\Phi_e(e) \downarrow$ implique $f(e) \neq \Phi_e(e)$. Notons qu'il s'agit alors d'une contradiction : si $f$ est calculable, alors il existe un code $e$ tel que $\Phi_e(n) \downarrow = f(n)$ pour tout $n$, et donc en particulier tel que $\Phi_e(e) \downarrow = f(e)$.

Les ensembles $\{\exists t \ \Phi_e(e)[t] \downarrow = 0\}$ et $\{\exists t \ \Phi_e(e)[t] \downarrow = 1\}$ sont c. e., et donc d'après le théorème 3.4 il existe $F_0(e)$ et $F_1(e)$ des formules $\Sigma_1$ telles que

$$\begin{aligned} \{e \in \mathbb{N} \ : \ \mathbb{N} \vDash F_0(e)\} &= \{e \in \mathbb{N} \ : \ \exists t \ \Phi_e(e)[t] \downarrow = 0\} \\ \{e \in \mathbb{N} \ : \ \mathbb{N} \vDash F_1(e)\} &= \{e \in \mathbb{N} \ : \ \exists t \ \Phi_e(e)[t] \downarrow = 1\}. \end{aligned}$$

Notons que PA démontre aussi $F_0(e) \to \neg F_1(e)$ et $F_1(e) \to \neg F_0(e)$ : autrement dit, $e \mapsto \Phi_e(e)$ est une fonction partielle qui ne peut avoir à la fois 0 et 1 comme valeur pour le même élément.

Pour calculer la valeur de $f(e)$, on énumère alors à l'aide du théorème 3.7 toutes les formules démontrées par $T$ jusqu'à tomber sur $F_0(e)$ ou $\neg F_0(e)$. Par hypothèse, une de ces deux éventualités arrive forcément. Si $T \vdash F_0(e)$, alors on définit $f(e) = 1$. Sinon, on définit $f(e) = 0$.
Montrons que notre fonction a la propriété attendue.
Si $\exists t\ \Phi_e(e)[t] \downarrow = 0$, alors $\mathbb{N} \vDash F_0(e)$, et donc d'après le lemme 3.8 $T \vdash F_0(e)$. Ainsi, $f(e) = 1 \neq \Phi_e(e)$.
Si à présent $\exists t\ \Phi_e(e)[t] \downarrow = 1$, alors $\mathbb{N} \vDash F_1(e)$, et l'on a donc d'après le lemme 3.8 $T \vdash F_1(e)$. On a alors $T \nvdash F_0(e)$, et donc $T \vdash \neg F_0(e)$. Ainsi, $f(e) = 0 \neq \Phi_e(e)$.
Enfin, si $\forall t\ \Phi_e(e)[t] \uparrow$, la valeur de $f$ n'importe pas. ∎

**Corollaire 3.11**
*Soit $T \supseteq$ PA une théorie complète et cohérente. Alors, $T$ calcule un ensemble $DNC_2$.*

Preuve. D'après la preuve du théorème précédent. ∎

Arrêtons-nous quelques instants sur ce que nous dit le théorème de Gödel-Rosser : il existe $F$ une formule $\Sigma_1$ qui n'est ni démontrable ni réfutable dans PA. D'après le lemme 3.8, si une formule $\Sigma_1$ est vraie dans $\mathbb{N}$, elle est démontrable dans PA. On en déduit $\mathbb{N} \nvDash F$, et donc $\mathbb{N} \vDash \neg F$ : cela fait de $\neg F$ une formule vraie, dans le sens où elle est vraie dans $\mathbb{N}$, mais que l'on ne peut démontrer à l'aide des axiomes de PA.

Le théorème nous dit enfin que rajouter des axiomes est sans espoir : tant que l'on garde la théorie calculatoirement énumérable, elle restera incomplète. Notons que l'on a montré avec la proposition 2.24 que $T$ pouvait tout à fait être étendue en une théorie complète et cohérente, mais ce sera au prix de ne plus en connaître les axiomes, et l'on serait alors bien en peine de l'utiliser pour démontrer quoi que ce soit...

Passons à présent au deuxième théorème de Gödel, plus surprenant encore que le premier, et qui mit un coup d'arrêt brutal au programme de Hilbert.

**Notation**

Pour une théorie calculatoirement énumérable $T$ dans $\mathcal{L}_{PA}$, soit $\mathrm{Coh}(T)$ la formule close $\Pi_1$ correspondant à « $T$ est une théorie cohérente », c'est-à-dire « pour toute preuve $p$ dans $T$, $p$ n'est pas une preuve de $\bot$ ».

L'existence d'une telle formule découle de la correspondance entre ensembles $\Sigma^0_n/\Pi^0_n$ et formules $\Sigma_n/\Pi_n$.

**Théorème 3.12 (Second théorème d'incomplétude de Gödel)**
*Soit $T$ une théorie cohérente calculatoirement énumérable contenant PA. Alors, $T \nvdash \mathrm{Coh}(T)$.*

Preuve. La première étape est de voir que la preuve du théorème 3.10 peut être formalisée, via un codage approprié, dans l'arithmétique de Peano : il existe $F$ une formule $\Sigma_1$ telle que

$$\mathrm{PA} \vdash \mathrm{Coh}(T) \to (\ulcorner T \nvdash F \urcorner \wedge \ulcorner T \nvdash \neg F \urcorner).$$

Les notations $\ulcorner \Psi \urcorner$ indiquent la transformation de $\Psi$ en un énoncé de l'arithmétique.

Supposons par l'absurde $T \vdash \mathrm{Coh}(T)$. Alors, par la règle du Modus Ponens, on a $T \vdash \ulcorner T \nvdash F \urcorner \wedge \ulcorner T \nvdash \neg F \urcorner$, et donc $T \vdash \ulcorner T \nvdash \neg F \urcorner$ ainsi que $T \vdash \ulcorner T \nvdash F \urcorner$.

Il faut ensuite constater que la preuve du lemme 3.8 également peut se faire dans l'arithmétique de Peano, c'est-à-dire que l'arithmétique de Peano montre que si une formule $\Sigma_1$ fixée est vraie, alors elle est démontrable dans l'arithmétique de Peano — et donc dans $T$. Cela donne donc formellement avec la formule $F$

$$T \vdash F \to \ulcorner T \vdash F \urcorner.$$

On a alors par contraposée (en utilisant l'équivalence entre la négation et le codage de la négation) :

$$T \vdash \ulcorner T \nvdash F \urcorner \to \neg F.$$

Comme $T \vdash \ulcorner T \nvdash F \urcorner$, alors par Modus Ponens on obtient

$$T \vdash \neg F.$$

Il suffit enfin de voir que si $T$ démontre une formule quelle qu'elle soit, alors PA — et donc $T$ — démontre que $T$ démontre cette formule. C'est encore une fois une application de la formalisation de la preuve du lemme 3.8 dans $T$, la phrase $T \vdash F$ étant $\Sigma_1$. On a donc finalement

$$T \vdash \ulcorner T \vdash \neg F \urcorner,$$

ce qui contredit $T \vdash \ulcorner T \nvdash \neg F \urcorner$. ∎

### 3.3. Conséquence des théorèmes d'incomplétude

Souvenons-nous du théorème, lui, de complétude de Gödel : $T \vdash F$ ssi tout modèle de $T$ est un modèle de $F$. Vu le second théorème d'incomplétude, aucune théorie $T$ cohérente et calculatoirement énumérable contenant l'arithmétique ne peut démontrer sa propre cohérence : $T \nvdash \mathrm{Coh}(T)$. On en déduit donc par exemple qu'il existe des modèles de l'arithmétique de Peano au sein desquels la phrase Coh(PA) est fausse : dans ces modèles, il existe en

particulier une démonstration de $0 = 1$. Nous avons pourtant montré que PA avait des modèles, et donc, d'après le théorème de cohérence, que PA ne pouvait pas démontrer $0 = 1$. Cette situation qui semble paradoxale est résolue avec la considération suivante : la preuve de $0 = 1$ dans un modèle de PA $\cup \{\neg\text{Coh(PA)}\}$ n'est pas une vraie preuve. La phrase Coh(PA) une fois exprimée dans le langage de l'arithmétique est de la forme : « il existe un entier qui code pour une démonstration valide de 0=1 dans PA ». Un tel modèle contiendra donc un tel entier, mais cet entier ne sera pas un véritable entier — sinon, en déroulant la démonstration codée par cet entier, on aurait une vraie démonstration de $0 = 1$.

Un tel modèle de PA est dit *non standard* : il comporte des entiers dits eux aussi non standard. Tout modèle de PA contient 0 et 1. Par les axiomes régissant l'addition, on montre aisément que tout modèle de PA contient bien sûr les entiers standard : $0, 1, 2, 3, 4, \ldots$ Un modèle non standard de PA contiendra des entiers *plus grands* que tous les entiers standard, et du point de vue du modèle, rien ne permet de distinguer ces entiers-là des autres.

Prenons un entier non standard $a$ d'un tel modèle. En utilisant les axiomes de l'arithmétique de Peano, on voit que les entiers $a + 1, a + 2, a + 3, \ldots$ existent également dans le modèle. Par l'axiome qui stipule que tout entier autre que 0 admet un prédécesseur, on voit aussi que les différents entiers $a - 1, a - 2, a - 3, \ldots$ sont dans le modèle. Comme $a > n$ pour tout $n \in \mathbb{N}$, alors également $a + a > a + n$ pour tout $n \in \mathbb{N}$. Il existe donc aussi un entier non standard plus grand que tous les $a + n$. En jouant ainsi avec les axiomes de PA, on arrive au théorème suivant.

**Théorème 3.13**
*L'ordre $<$ pour les modèles dénombrables et non standard de l'arithmétique est constitué d'une copie de $\mathbb{N}$, suivie de $\mathbb{Q}$ copies de $\mathbb{Z}$.*

On connaît donc bien à quoi ressemble l'ordre des éléments dans un modèle non standard. On ne connaît en revanche malheureusement pas à quoi ressemblent ni l'addition ni la multiplication.

**Théorème 3.14 (Tennenbaum)**
*Soit $\mathcal{M}$ un modèle non standard dénombrable de PA. Soit*

$$+, \times : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

*des fonctions qui représentent les fonctions d'addition et de multiplication de $\mathcal{M}$, une fois établie une bijection entre $\mathcal{M}$ et $\mathbb{N}$. Alors, ni $+$ ni $\times$ ne sont calculables.*

Nous reparlerons un peu des modèles non standard de l'arithmétique dans l'étude des mathématiques à rebours, et notamment dans la section 23-3.

# 4. Système ZFC

Nous avons donné une preuve de la cohérence de l'arithmétique de Peano, en utilisant le théorème de cohérence (voir le corollaire 2.21) et en fournissant un modèle pour cette théorie. D'après le deuxième théorème d'incomplétude de Gödel, nous avons nécessairement utilisé une théorie plus puissante que PA pour créer ce modèle. De quelle théorie s'agit-il ? On peut donner plusieurs réponses à cette question. La théorie la plus naturelle permettant de montrer la consistance de l'arithmétique du premier ordre est sans doute la théorie de l'arithmétique *du second ordre*, qui sera abordée plus en détail dans le chapitre 22.

## 4.1. Motivations

En arithmétique du second ordre, on s'autorise non seulement à travailler avec des entiers, mais également avec des ensembles d'entiers. L'existence d'ensembles d'entiers arbitraires est sujette à caution, davantage en tout cas que l'existence des entiers eux-mêmes : il s'agit d'objets infinis, qui sont en quantité indénombrable, et s'il y a un aspect « légitime » à accepter l'existence d'ensembles d'entiers calculables (après tout, on peut écrire des algorithmes qui les produisent, et leur existence théorique est donc doublée d'une certaine forme d'existence pratique), on peut questionner plus facilement celle des autres ensembles. Certains d'entre eux sont malgré tout accessibles, dans le sens où ils ont une définition claire, par exemple le saut Turing. Ce qui rend $\emptyset'$ plus légitime qu'un autre ensemble non calculable arbitraire, c'est le fait qu'il soit *définissable*, et qui plus est par une formule assez simple — en particulier $\Sigma_1$ : $\{e \in \mathbb{N} \ : \ \Phi_e(e) \downarrow\}$. En arithmétique du second ordre, on s'autorisera l'existence de tous les ensembles définissables par une formule de l'arithmétique arbitraire, en particulier les formules $\Sigma^0_n$ pour un certain $n$ : cela s'appelle *l'axiome de compréhension*. Une fois que l'on a admis l'existence d'un ensemble $X$, il serait absurde de ne pas accepter l'existence d'ensembles calculables avec $X$ comme oracle, et si l'on accepte l'axiome de compréhension, il serait tout aussi absurde de ne pas accepter l'existence des ensembles définissables par une formule de l'arithmétique arbitraire, qui pourrait utiliser $X$ comme oracle.

L'arithmétique du second ordre consiste donc en l'arithmétique de Peano, à laquelle on ajoute l'axiome de compréhension qui permet de valider l'existence des ensembles définissables via une formule de l'arithmétique, éventuellement à l'aide d'un oracle — un ensemble déjà existant. Une très grande partie des mathématiques peut déjà se formaliser de cette manière, mais pas toutes les mathématiques. En particulier, rien ne permet dans

l'arithmétique du second ordre de parler de l'ensemble de tous les sous-ensembles d'entiers, ou même de sous-ensembles arbitraires de ce dernier. On a besoin pour cela d'un autre axiome : étant donné un ensemble $X$, l'ensemble $\mathcal{P}(X)$ des parties de $X$ —c'est-à-dire l'ensemble de tous les sous-ensembles de $X$ — est légitime, il existe et on peut l'utiliser. L'ensemble des parties de $X$ n'est pas de même nature que $X$. L'un est un ensemble d'entiers et l'autre un ensemble d'ensembles d'entiers. La théorie des ensembles permet de traiter ces deux éléments de manière homogène : tout élément ne sera plus qu'ensemble, y compris les entiers. Informellement, l'entier 0 sera l'ensemble vide, l'entier 1 sera l'ensemble qui contient 0, l'entier 2 sera l'ensemble qui contient 0 et 1, et inductivement l'ensemble $n+1$ sera l'ensemble qui contient $m$ pour tout $m \leqslant n$. On peut bien sûr imaginer d'autres manières de représenter les entiers par des ensembles, et avec la pratique ces dernières n'ont aucune importance, mais il faut bien en choisir une, et cette manière-là est celle qui est devenue standard, sous l'impulsion du mathématicien John von Neumann. Nous en reparlerons avec l'étude des ordinaux dans le chapitre 27.

### 4.2. Système de Zermelo

À présent que l'on ne travaille plus qu'avec des ensembles, nous ne sommes plus dans le langage de l'arithmétique. Notre unique symbole de relation est celui de l'appartenance $\in$ — auquel on ajoute tout de même celui de l'égalité. Il nous faut quelques axiomes basiques afin de régir les manipulations élémentaires des ensembles.

(1) *L'axiome de l'ensemble vide* : l'ensemble vide existe.

(2) *L'axiome de paire* : si $a$ et $b$ sont des ensembles, alors $\{a, b\}$ est un ensemble.

(3) *L'axiome de réunion* : l'idée est que si $(a_i)_{i\in I}$ est une collection d'ensembles indexée par un ensemble $I$, alors l'ensemble $\bigcup_{i\in I} a_i$ existe. La notion « indexée » n'étant pas formellement définie en théorie des ensembles, on dira à la place que si $b$ est un ensemble dont les éléments sont les $a_i$, alors la réunion de tous ces $a_i$ existe. Pour être tout à fait formel, l'axiome est : $\forall b\ \exists c\ \forall x\ (x \in c \leftrightarrow \exists a \in b\ x \in a)$.

On a également besoin d'un axiome qui régisse l'égalité entre deux ensembles.

(4) *L'axiome d'extensionnalité* : deux ensembles sont égaux si, et seulement si, ils ont les mêmes éléments.

Il y a finalement notre axiome fauteur de troubles, qui nous a poussé dans cette théorie des ensembles permettant — par exemple — de traiter de l'ensemble des parties de $\mathbb{N}$.

(5) *L'axiome de l'ensemble des parties* : pour tout ensemble $a$, l'ensemble des parties de $a$, noté $P(a)$ existe. Formellement :

$$\forall a\ \exists b\ \forall c\ (c \in b \leftrightarrow c \subseteq a),$$

où $c \subseteq a$ peut s'écrire comme $\forall x(x \in c \to x \in a)$.

Nous pouvons enfin ajouter notre axiome de compréhension, permettant de construire des ensembles à partir de formules du premier ordre utilisant éventuellement d'autres ensembles comme paramètres. Comme il existe une infinité de formules du premier ordre, il ne s'agit pas d'un seul axiome, mais d'un schéma d'axiome : un axiome par formule.

(6) *Le schéma d'axiome de compréhension* : pour $F(y, x_1, \ldots, x_n)$, formule fixée dans le langage de la théorie des ensembles, pour tout $n$-uplet d'ensembles $b_1, \ldots, b_n$ et pour tout ensemble $a$, l'ensemble

$$\{y \in a\ :\ F(y, b_1, \ldots, b_n)\}$$

existe.

On peut vérifier sans peine que les axiomes précédents impliquent l'existence de tous les ensembles héréditairement finis, c'est-à-dire des ensembles finis, dont les éléments sont eux-mêmes des ensembles finis, etc., jusqu'à arriver en déroulant l'arbre décrivant les relations d'appartenance d'un ensemble avec ses éléments, à l'ensemble vide pour toute feuille de cet arbre. Rien ne nous permet pour le moment de parler de l'ensemble de tous les entiers. Nous avons de fait besoin pour cela d'un axiome.

(7) *L'axiome de l'infini* : il existe un ensemble infini. Formellement :

$$\exists x\ (\emptyset \in x \wedge \forall y \in x\ \ y \cup \{y\} \in x).$$

L'axiome de l'infini asserte moralement l'existence de $\mathbb{N}$ en tant qu'ensemble. Via le codage que nous avons donné des entiers, on peut vérifier que l'ensemble codant l'entier $n+1$ est égal à l'ensemble $n \cup \{n\}$, où $n$ est l'ensemble codant l'entier $n$. L'axiome de l'infini nous dit donc qu'il existe un ensemble contenant tous les entiers. Il pourrait éventuellement contenir d'autres éléments, mais en utilisant les autres axiomes on définit $\mathbb{N}$ comme le plus petit ensemble $x$ — plus petit pour l'inclusion — tel que

$$\emptyset \in x \wedge \forall y \in x\ \ y \cup \{y\} \in x.$$

### 4.3. L'axiome de remplacement et les jeux boréliens

Les axiomes de (1) à (7) forment la théorie Z de Zermelo. Ils sont suffisants pour développer une large partie des mathématiques. Fraenkel et Skolem vont introduire un nouvel axiome, à la fois intuitif et formellement

nécessaire pour développer les théories des ordinaux et des hiérarchies d'infinis. Il s'agit plus exactement un schéma d'axiomes, qui dit essentiellement que l'image d'une formule fonctionnelle existe. Une formule $F(y, r)$ est fonctionnelle si pour tout $y$ la formule $F(y, r_y)$ est satisfaite pour exactement un ensemble $r_y$. Il s'énonce formellement comme suit.

(8) *Le schéma d'axiome de remplacement* : pour une formule fonctionnelle $F(y, x_1, \dots x_n, z)$ fixée dans le langage de la théorie des ensembles, pour tout $n$-uplet d'ensembles $b_1, \dots, b_n$, pour tout ensemble $a$, l'ensemble
$$\{z \ : \ \exists y \in a \ F(y, b_1, \dots, b_n, z)\}$$
existe.

La théorie des ensembles devient strictement plus puissante avec le schéma d'axiome de remplacement, qui permet en particulier de montrer l'existence de l'ensemble $\mathbb{N} \cup P(\mathbb{N}) \cup P(P(\mathbb{N})) \dots$, dont la construction est impossible sans cet axiome.

Il est remarquable de noter qu'une utilisation conjointe de l'axiome de l'ensemble des parties et du schéma d'axiome de remplacement est indispensable pour construire certains ensembles d'entiers — qui seront nécessairement d'une complexité extrême en termes de degré Turing. L'exemple emblématique est le théorème de détermination des jeux boréliens de Martin. Voyons de quoi il s'agit : considérons une classe $\mathcal{B} \subseteq 2^{\mathbb{N}}$ pour le moment arbitraire, et considérons le jeu à deux joueurs suivant : le joueur 1 choisit un bit $x_0 \in \{0, 1\}$, puis le joueur 2 choisit à son tour un bit $x_1 \in \{0, 1\}$, et ainsi de suite, à l'étape $2n$ le joueur 1 choisit le bit $x_{2n}$ et à l'étape $2n + 1$ le joueur 2 choisit le bit $x_{2n+1}$. À « la fin » du jeu, on obtient un ensemble $X = x_0 x_1 x_2 \dots$ Le joueur 1 gagne le jeu si $X \in \mathcal{B}$, sinon c'est le joueur 2 qui remporte la partie. La question est alors : un des deux joueurs a-t-il une stratégie gagnante ? Une stratégie pour le joueur 1 est une fonction $f$ qui prend en paramètre une chaîne $\sigma$ de taille paire, correspondant à ce qui a été joué jusque-là, le dernier coup étant le dernier bit de $\sigma$ joué par le joueur 2, et qui renvoie le coup suivant $f(\sigma)$. Une telle stratégie est gagnante si l'ensemble obtenu en jouant les coups donnés par la fonction $f$ est toujours dans $\mathcal{B}$, quels que soient les coups joués par le joueur 2.

Nous verrons dans le chapitre 17 des classes $\mathcal{B} \subseteq 2^{\mathbb{N}}$ — d'une variété fort riche — ne nécessitant pas l'axiome de l'ensemble des parties pour être manipulées : nous en avons déjà un exemple avec les classes $\Pi^0_1$ ou $\Sigma^0_1$. Nous mènerons des constructions itérées de classes de plus en plus complexes, qui peuvent se coder par un objet dénombrable, d'une manière analogue au codage des classes $\Pi^0_1$ par des arbres : il s'agira des classes dites *boréliennes*. Le mathématicien Donald A. Martin, dont nous reparlerons dans le chapitre 12, a montré le remarquable théorème suivant.

**Théorème 4.1 (Martin [149])**
*Soit $\mathcal{B}$ une classe borélienne. Alors, pour le jeu décrit ci-dessus avec la classe $\mathcal{B}$, un des deux joueurs a une stratégie gagnante.*

La stratégie d'un jeu borélien est une fonction $f : 2^{<\mathbb{N}} \to \{0, 1\}$, qui peut donc être représentée par un ensemble d'entiers. Friedman [67] a montré que pour certains boréliens, de complexité relativement simple, une telle stratégie ne pouvait être construite sans l'utilisation de l'axiome de l'ensemble des parties, et même sans itération arbitraire de l'application de cet axiome. Ainsi, pour montrer l'existence de certains réels, nous faut-il faire appel à l'axiome de l'ensemble des parties utilisé conjointement à celui de remplacement afin de construire $\mathbb{N} \cup P(\mathbb{N}) \cup P(P(\mathbb{N})) \ldots$, indispensable à la définition de notre réel — la stratégie gagnante pour un certain borélien.

### 4.4. L'axiome de fondation et le codage

Un autre axiome a été rajouté par Fraenkel et Skolem, ainsi que par von Neumann. Contrairement aux autres axiomes, ce dernier est à peu près inutile pour la construction de l'univers mathématique, mais on l'ajoute simplement parce que cela correspond à notre conception des choses : l'axiome dit en substance qu'étant donné un ensemble $x$, si l'on considère un élément $x_1 \in x$, puis un élément $x_2 \in x_1$, en continuant ainsi inductivement avec un élément $x_{n+1} \in x_n$, on arrivera nécessairement au bout d'un certain $n \in \mathbb{N}$ à l'ensemble vide : il n'y a pas d'autres ensembles que ceux que l'on peut construire inductivement via les autres axiomes en partant de l'ensemble vide. En particulier, il n'y a aucun ensemble $x$ tel que $x \in x$. Cela peut paraître évident ou pas selon chacun, mais reflète en tout cas la conception généralement adoptée dans la communauté sur la nature des ensembles.

Afin d'étayer notre propos, rappelons que l'un des intérêts de la théorie des ensembles est de pouvoir y formaliser la totalité des mathématiques. Cette formalisation passe par le codage des structures mathématiques usuelles par des ensembles, et ce codage ne se fait de toute façon qu'avec des ensembles qui respectent l'axiome de fondation — que ce dernier soit adopté ou pas. Aussi, s'il n'est pas contradictoire de penser qu'il existe d'autres ensembles qui ne respectent pas cet axiome, on n'en a en pratique pas besoin, et de tels objets ne correspondent *a priori* à rien de tangible.

Terminons en insistant sur le fait que si l'on peut coder le reste des mathématiques par des ensembles, ce n'est en revanche pas une raison pour le faire, et l'on est évidemment bien plus à l'aise à travailler avec des entiers, des réels ou autre. Ce qui est intéressant c'est l'*existence* de ce codage, et le fait que si un énoncé est indécidable en théorie des ensembles, il en devient alors indécidable « tout court ».

### 4.5. L'axiome du choix et la cardinalité

La théorie Z de Zermelo plus l'axiome de remplacement et celui de fondation donne la théorie ZF, de Zermelo/Fraenkel.

Un dernier axiome, sans doute le plus célèbre, est l'axiome du choix, qui faisait originellement partie de la théorie de Zermelo. La nécessité de cet axiome devrait être aisée à comprendre via l'analogie que l'on peut en faire en calculabilité. Nous avons par exemple vu que toute classe $\Pi^0_1$ non vide et dénombrable contient un ensemble calculable. En revanche, il n'est pas possible, étant donné le code d'une classe $\Pi^0_1$ non vide, de trouver uniformément un algorithme permettant d'en calculer un élément. En particulier, étant donné une suite $(\mathcal{F}_n)_{n\in\mathbb{N}}$ de classes $\Pi^0_1$ dénombrables non vides, il n'existe pas nécessairement de fonction calculable permettant de *choisir* un élément dans chacune de ces classes. La question clef est ici celle de l'uniformité. Évidemment, à l'aide de $\emptyset'$, on pourra créer cette fonction de choix, mais que se passe-t-il si l'on considère des classes plus complexes ? Peut-on toujours construire une fonction de choix ? La réponse est non, sans l'aide d'un nouvel axiome.

(9) *L'axiome du choix* : étant donné une collection $(A_i)_{i\in I}$ d'ensembles non vides, il existe une fonction $f : I \to \bigcup_{i\in I} A_i$ telle que $f(i) \in A_i$ pour tout $i$.

L'axiome du choix apparaît nécessaire pour développer une théorie complète de la cardinalité des ensembles. En particulier, avec l'axiome du choix, on peut montrer que pour tous ensembles $A, B$ on a $|A| \leqslant |B|$ ou $|B| \leqslant |A|$ (nous en reparlerons avec l'étude détaillée des ordinaux dans le chapitre 27). Ce n'est plus du tout vrai sans l'axiome du choix, l'exemple par excellence en calculabilité étant certainement celui des degrés Turing. Il est facile de construire une injection de $2^{\mathbb{N}}$ dans les degrés Turing, mais il est en revanche impossible de construire une injection des degrés Turing dans $2^{\mathbb{N}}$ sans l'axiome du choix (il est en particulier impossible de montrer l'existence d'une fonction $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ telle que $X \equiv_T Y \leftrightarrow f(X) = f(Y)$).

Une première réaction est de se simplifier la vie et d'utiliser l'axiome du choix si nécessaire. Cette approche n'est toutefois pas dans l'esprit de la calculabilité, de laquelle la théorie des ensembles est moins éloignée que l'on ne pourrait le croire : les axiomes autres que l'axiome du choix nous permettent de *construire* des objets de plus en plus complexes à partir d'objets existants, un peu à la manière dont on construit des degrés Turing de plus en plus complexes en itérant le saut. L'axiome du choix est quant à lui fondamentalement non constructif, et n'est de ce point de vue là pas aussi légitime que les autres. Il conduit en plus à des théorèmes qui semblent paradoxaux, l'exemple le plus connu étant le *paradoxe de Banach/Tarski*, une

construction qui à l'aide de l'axiome du choix montre comment découper une boule de l'espace $\mathbb{R}^3$ en un nombre fini de morceaux, et comment réassembler ces moreaux pour obtenir deux boules strictement identiques à la première.

En n'acceptant pas l'axiome du choix, on sort bien entendu de ce monde idéal où la cardinalité de tout ensemble est comparable, mais il s'agit en fait d'une situation « artificielle », dont nous n'avons pas réellement besoin.

### 4.6. Résultats d'indépendances

La théorie obtenue avec les axiomes (1)-(8) est la théorie ZF, et si on lui ajoute l'axiome du choix, on a alors la théorie dite ZFC.

#### 4.6.1. L'axiome du choix

La question de savoir si l'axiome du choix est démontrable à partir des autres axiomes, ou même celle de savoir s'il ne risque pas d'introduire de contradiction, est longtemps restée ouverte. Le théorème de complétude de Gödel permet la technique suivante : si l'on fait l'hypothèse que ZF est cohérent, et donc a un modèle, et qu'à l'aide de ce modèle on peut construire un modèle de ZFC, on aura montré que la cohérence de ZF implique celle de ZFC, et en particulier que ZF ne peut pas démontrer que l'axiome du choix est faux (à moins bien sûr que ZF ne soit incohérent). C'est exactement ce qu'a fait Gödel quelques années plus tard [74], via son modèle dit de *l'univers des constructibles*. Ce n'est qu'encore plus tard, en 1962, avec sa fameuse technique de Forcing dont nous verrons certains aspects dans le chapitre 11, que Cohen [37] réussira le tour de force de construire un modèle de ZF dans lequel l'axiome du choix est faux : l'axiome du choix ne peut donc être ni démontré ni réfuté dans ZF.

#### 4.6.2. L'hypothèse du continu

La question qui obséda Cantor tout au long de sa vie (et de nombreux mathématiciens durant presque un siècle) est elle aussi indépendante des autres axiomes de la théorie des ensembles (avec ou sans l'axiome du choix). L'univers des constructibles de Gödel constitue également un modèle de ZFC dans lequel l'hypothèse du continu est vérifiée, c'est-à-dire qu'il n'existe aucun ensemble $A$ pour lequel $|\mathbb{N}| < |A| < |2^{\mathbb{N}}|$. Toujours avec sa technique de forcing, Cohen a construit des modèles de ZFC dans lesquels on a un nombre arbitraire d'infinis strictement compris entre $|\mathbb{N}|$ et $|2^{\mathbb{N}}|$.

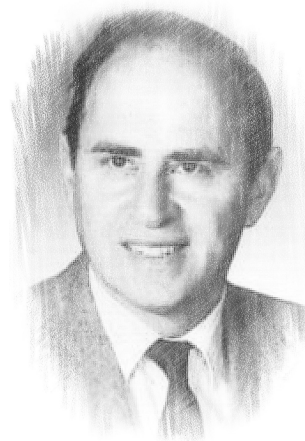
Notons ici que l'on peut donner plusieurs versions de l'hypothèse du continu. Celle formulée originellement par Cantor portait sur les ensembles de réels : existe-t-il un ensemble $\mathcal{A} \subseteq \mathbb{R}$ tel que $|\mathbb{N}| < |A| < |\mathbb{R}|$ ? Plus tard, la question sera étendue à n'importe quel ensemble, et en particulier aux ordinaux,

que nous verrons formellement dans le chapitre 26. Quelle que soit la version de l'hypothèse du continu, il s'agit d'une question indépendante des autres axiomes de la théorie des ensembles.

Chapitre 10

# Forcing de Cohen

Le forcing est une technique inventée par le mathématicien Paul Cohen au début des années 60 pour montrer l'indépendance de l'hypothèse du continu et de l'axiome de choix de la théorie de Zermelo/Fraenkel. Cette technique a révolutionné la théorie des ensembles, et joue également un rôle prépondérant en calculabilité, où elle s'est imposée comme l'une des deux techniques principales de construction d'ensembles, aux côtés de la méthode de priorité (voir le chapitre 13).



Paul Joseph Cohen, 1934–2007

Historiquement, cette technique a été conçue pour étendre un modèle $\mathcal{M}$ de la théorie de ZF en lui ajoutant un nouvel objet $G$ pour former un nouveau modèle $\mathcal{M}[G]$ tout en permettant aux éléments du modèle $\mathcal{M}$ de contrôler les propriétés du modèle étendu $\mathcal{M}[G]$. Plus généralement, la technique de forcing permet de créer des objets mathématiques à l'aide d'approximations de plus en plus précises, tout en étant capable de contrôler certaines propriétés de l'objet final avant que la construction ne soit terminée. Une notion de forcing est avant tout la définition d'un ordre partiel d'approximations $(\mathbb{P}, \leqslant)$, où la relation $d \leqslant c$ signifie que l'approximation $d$ est plus précise que l'approximation $c$. Lorsqu'une propriété (ou contrat) $\mathcal{R}$ sur l'objet final est déjà déterminée par

une approximation $c \in \mathbb{P}$, autrement dit lorsque quelle que soit la suite de la construction, l'objet final satisfera $\mathcal{R}$, on dit alors que *c force* $\mathcal{R}$. Le cœur de la technique du forcing réside dans la capacité à forcer des propriétés complexes sur l'objet final à partir de simples approximations.

Le forcing est souvent considéré comme une technique difficile à appréhender aux premiers abords. Son application à la calculabilité, plus simple, permet d'une part de l'aborder en douceur, et d'autre part de comprendre plus facilement le détail de ses mécanismes sous-jacents. Cette simplicité est due essentiellement aux deux choses suivantes.

- ▷ À l'instar de la méthode des extensions finies, nous allons principalement nous servir du forcing pour créer des ensembles d'entiers, tout en contrôlant leurs puissances calculatoires. On s'intéressera donc à forcer des formules simples, du type $\ll \Phi^G(n)\downarrow \gg$ ou $\ll \Phi^G(n)\uparrow \gg$, où $G$ désigne l'ensemble final. Il s'agit donc de forcer des formules $\Sigma^0_1$ ou $\Pi^0_1$. Nous verrons que plusieurs constructions, notamment celles avec la méthode des extensions finies, sont des utilisations simplifiées du forcing. Nous verrons également comment étendre le forcing à des formules de complexité arbitraires, ce qui présente un premier niveau de complexité supplémentaire.
- ▷ Contrairement à la théorie des ensembles, nous ne forcerons que des propriétés dont les quantificateurs portent sur les entiers. La création d'un ensemble d'entiers par forcing ne modifie pas la nature des entiers, et donc la portée des quantificateurs. En théorie des ensembles en revanche, les quantificateurs portent sur les ensembles, et la création d'un nouvel ensemble ajoute une grande quantité d'ensembles au modèle, et change donc la portée des quantificateurs. Il est alors nécessaire d'utiliser des *noms*, pour parler des objets de notre modèle étendu à l'intérieur de notre modèle de base. Nous n'aurons pas besoin d'avoir recours aux noms en calculabilité, ce qui rend la présentation plus abordable.

## 1. Formules de l'arithmétique du second ordre

Les énoncés que l'on « force » en calculabilité sont toujours des formules de l'arithmétique du premier ordre *avec des variables libres d'ensembles*. Ces formules sont un cas particulier de l'arithmétique du second ordre, dont nous parlerons plus en détail dans le chapitre 22. Une formule de l'arithmétique du second ordre possède deux types de variables : des variables dites *du premier ordre* représentant des entiers et que l'on notera en minuscule, et des variables dites *du second ordre* représentant des ensembles d'entiers et que l'on notera en majuscule. Le langage se voit augmenté du symbole d'appartenance $\in$, et de la formule atomique

« $x \in A$ ». En arithmétique du second ordre, les quantifications peuvent être sur les entiers, et sur les ensembles d'entiers. Par exemple, l'énoncé « $\forall A\ \forall n\ \exists m\ (m > n \wedge m \in A)$ » affirme que tout ensemble d'entiers est infini.

L'arithmétique du premier ordre avec variables libres d'ensembles est une restriction de l'arithmétique du second ordre, où seules les quantifications sur les entiers sont autorisées [1]. Lors de l'évaluation d'une formule de l'arithmétique du second ordre dans un modèle donné, ses variables libres d'ensembles sont remplacées par des paramètres, c'est-à-dire des éléments du modèle considéré. Ainsi, l'énoncé $F(G) = \forall n\ \exists m\ (m > n \wedge m \in G)$ est une formule de l'arithmétique du premier ordre avec $G$ comme variables libres d'ensemble, et tout ensemble infini $X \subseteq \mathbb{N}$ est un paramètre pour lequel $F(X)$ est vrai.

Nous avons vu dans la section 9-3 les formules $\Delta_0^0$ de l'arithmétique : celles ne contenant que des quantifications bornées, c'est-à-dire des quantifications de la forme $\forall x \leqslant y$ et $\exists x \leqslant y$. Nous pouvons définir une hiérarchie sur les formules de l'arithmétique avec variables libres d'ensembles, similaire à la hiérarchie de la définition 9-3.2.

**Définition 1.1**

1. Une formule $F(Y_1, \ldots, Y_k, x_1, \ldots, x_m)$ de l'arithmétique du second ordre est $\Sigma_n^0$ si
$$F(Y_1, \ldots, Y_k, x_1, \ldots, x_m) = \overbrace{\exists y_1 \forall y_2 \ldots Q y_n}^{n \text{ quantificateurs}} G(Y_1, \ldots, Y_k, x_1, \ldots, x_m, y_1, \ldots, y_n),$$
pour $G(Y_1, \ldots, Y_k, x_1, \ldots, x_m, y_1, \ldots, y_n)$ une formule $\Delta_0^0$, où $Q$ vaut $\exists$ si $n$ est impair, et $\forall$ si $n$ est pair.
2. Une formule $F(Y_1, \ldots, Y_k, x_1, \ldots, x_m)$ de l'arithmétique du second ordre est $\Pi_n^0$ si
$$F(Y_1, \ldots, Y_k, x_1, \ldots, x_m) = \overbrace{\forall y_1 \exists y_2 \ldots Q y_n}^{n \text{ quantificateurs}} G(Y_1, \ldots, Y_k, x_1, \ldots, x_m, y_1, \ldots, y_n),$$
pour $G(Y_1, \ldots, Y_k, x_1, \ldots, x_m, y_1, \ldots, y_n)$ une formule $\Delta_0^0$, où $Q$ vaut $\forall$ si $n$ est impair, et $\exists$ si $n$ est pair. ♢

Nous avons jusqu'ici implicitement utilisé les formules de l'arithmétique du second ordre, via l'utilisation de fonctionnelles. Le théorème 9-3.4 se décline en l'équivalence suivante.

1. Nous ne verrons que très tard dans cet ouvrage, dans la partie IV, la grande complexité calculatoire qui se cache derrière les quantifications du second ordre.

**Théorème 1.2**
*Soient $A, Z \in 2^{\mathbb{N}}$. Les énoncés suivants sont équivalents.*

*(1)* $A \subseteq \mathbb{N}$ *est* $Z$*-c. e.*

*(2) Il existe* $F(X, n)$ *une formule* $\Sigma^0_1$ *de l'arithmétique du second ordre telle que* $A = \{n \in \mathbb{N} \ : \ \mathbb{N} \models F(Z, n)\}$.

*(3) Il existe une fonctionnelle Turing* $\Phi(Z, n)$ *telle que*

$$A = \{n \in \mathbb{N} \ : \ \Phi(Z, n)\downarrow\}.$$

## 2. Forcing $\mathbf{\Sigma^0_1/\Pi^0_1}$

Nous allons maintenant commencer notre immersion dans l'univers du forcing en étudiant une notion de forcing spécifique, à savoir le forcing de Cohen. Comme expliqué dans l'introduction, une notion de forcing est spécifiée par la donnée de son ordre partiel d'approximations. Dans notre cas, il s'agira de l'ordre partiel des chaînes, muni de la relation de préfixe. Il s'agit d'une des notions de forcing les plus simples conceptuellement, mais qui contient déjà les concepts fondamentaux du forcing.

Il existe plusieurs manières d'aborder le forcing, avec différents niveaux d'abstraction. Il est possible de le voir comme une élaboration de la méthode des extensions finies, cherchant à systématiser la satisfaction de contrats, et à en extraire une construction générale. Nous présenterons cette approche dans la section 2.1.

Il est également possible de formuler le forcing dans un cadre topologique. La topologie permet de définir une notion de classe d'ensembles « négligeable », ou *maigre*. La construction d'un ensemble par forcing consiste alors à choisir un élément « typique », c'est-à-dire évitant un ensemble négligeable de propriétés indésirables. Nous verrons cette approche dans la section 2.2.

### 2.1. L'approche par extensions finies

Reconsidérons à présent la méthode des extensions finies développée dans la section 4-8. Le but est de construire un ensemble $G \in 2^{\mathbb{N}}$ satisfaisant une infinité de contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$. Chaque contrat est traité indépendamment, et doit donc être satisfait tout en laissant suffisamment de degrés de liberté dans la construction pour satisfaire les autres contrats.

**Point sur les contrats**

Les contrats — que nous avons vus jusqu'ici de manière informelle — peuvent être vus comme des formules dans le langage de l'arithmétique du premier ordre avec variables d'ensembles, comportant une variable libre du second ordre représentant l'ensemble $G$ qui doit satisfaire le contrat. La méthode des extensions finies consiste à construire $G \in 2^{\mathbb{N}}$ en spécifiant des segments initiaux de plus en plus longs, représentés sous forme de chaînes $\sigma \in 2^{<\mathbb{N}}$. Les contrats $\Sigma^0_1$ sont ceux correspondant à des formules $\Sigma^0_1$ ou de manière équivalente ceux que l'on peut toujours mettre sous la forme $\ll \Phi_e(G,0)\downarrow \gg$ pour une fonctionnelle $\Phi_e$. Les contrats $\Pi^0_1$ sont ceux correspondant à des formules $\Pi^0_1$ ou de manière équivalente ceux que l'on peut toujours mettre sous la forme $\ll \Phi_e(G,0)\uparrow \gg$ pour une fonctionnelle $\Phi_e$.

**Satisfaction d'un contrat.** La procédure générale pour satisfaire un contrat $\mathcal{R}_e$ est la suivante : étant donné une chaîne $\sigma \in 2^{<\mathbb{N}}$ représentant un segment initial de $A$ déjà spécifié pour satisfaire les contrats précédents, il s'agit de trouver une chaîne $\tau \succeq \sigma$ telle que le contrat $\mathcal{R}_e$ est satisfait. L'ensemble final $G$ n'étant pas alors connu, le contrat doit être satisfait quelle que soit la suite de la construction, autrement dit il doit être satisfait pour tout $G \in [\tau] = \{X \in 2^{\mathbb{N}} : \tau \prec X\}$.

**Ordre partiel des chaînes.** Prenons de la hauteur, et considérons la méthode des extensions finies d'un point de vue plus abstrait. Nous avons un ordre partiel $(2^{<\mathbb{N}}, \preceq)$ qui correspond à l'ensemble $2^{<\mathbb{N}}$ des chaînes finies, muni de la relation de préfixe. Nous avons aussi une fonction d'interprétation $[\cdot] : 2^{<\mathbb{N}} \to \mathcal{P}(2^{\mathbb{N}})$ définie par $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \prec X\}$. Intuitivement, les éléments de $2^{<\mathbb{N}}$ représentent des approximations de l'ensemble $G$ en construction. Étant donné une approximation $\sigma$, $[\sigma]$ est la classe des ensembles que l'on pourrait potentiellement obtenir à la fin de la construction. Plus on avance dans la construction, plus l'approximation s'affine et la classe des ensembles candidats se restreint. Ainsi, nous avons la propriété suivante : si $\sigma \preceq \tau$, alors $[\tau] \subseteq [\sigma]$. Voyons une première définition de la relation de forcing.

**Définition 2.1.** Soit $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$. Une chaîne $\sigma$ *force* $\mathcal{R}$, que l'on notera par $\sigma \Vdash^* \mathcal{R}$, si le contrat est satisfait pour tout $G \in [\sigma]$. ♢

**Densité.** Nous pouvons nous abstraire de la notion de contrat en représentant un contrat $\mathcal{R}_e$ comme l'ensemble $P_e \subseteq 2^{<\mathbb{N}}$ des chaînes qui le forcent. Notons que si $\sigma$ force $\mathcal{R}_e$, alors tout $\tau \succeq \sigma$ force également $\mathcal{R}_e$, car $[\tau] \subseteq [\sigma]$. L'ensemble $P_e$ est donc clos par suffixe. La procédure de satisfaction du contrat $\mathcal{R}_e$ consiste à montrer que pour toute chaîne $\sigma \in 2^{<\mathbb{N}}$,

il existe une extension $\tau \succeq \sigma$ telle que $\tau \in P_e$. Si c'est le cas, on dira que l'ensemble $P_e$ est dense.

**Définition 2.2.** Un ensemble $W \subseteq 2^{<\mathbb{N}}$ est *dense* si pour tout $\sigma \in 2^{<\mathbb{N}}$, il existe une extension $\tau \succeq \sigma$ telle que $\tau \in W$. ♢

Intuitivement, un ensemble $W \subseteq 2^{<\mathbb{N}}$ est dense si quelle que soit la construction en cours $\sigma_0 \preceq \sigma_1 \preceq \cdots \preceq \sigma_n$ à l'aide de la méthode des extensions finies, il n'est jamais trop tard pour trouver une extension $\sigma_{n+1} \succeq \sigma_n$ dans $W$. Il s'ensuit que si nous avons un ensemble dénombrable de contrats représentés par des ensembles $(P_n)_{n\in\mathbb{N}}$, dès lors que ces ensembles sont denses, il existe une suite infinie $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ telle que pour tout $n$, il existe un entier $m$ pour lequel $\sigma_m \in P_n$. En particulier, soit $\{G\} = \bigcap_n [\sigma_n]$, alors $G$ possèdera des segments initiaux dans chaque ensemble $P_e$.

**Remarque**

Si les ensembles $(P_n)_{n\in\mathbb{N}}$ sont uniformément c.e, alors la construction de la suite infinie $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ peut se faire de manière calculable, et l'ensemble $G$ résultant l'est également.

**Généricité.** Nous pouvons formaliser la construction de la méthode des extensions finies en un théorème trivial au vu des intuitions précédentes.

**Définition 2.3.** On dit qu'un ensemble $G \in 2^{\mathbb{N}}$ *rencontre* un ensemble $W \subseteq 2^{<\mathbb{N}}$ si $G\restriction_n \ \in W$ pour un certain $n \in \mathbb{N}$. Soit $\vec{D} = (D_n)_{n\in\mathbb{N}}$ une suite d'ensembles de chaînes ; un ensemble $G \in 2^{\mathbb{N}}$ est *$\vec{D}$-générique* s'il rencontre chaque $D_n$. ♢

**Théorème 2.4**
*Soit $\vec{D} = (D_n)_{n\in\mathbb{N}}$ une suite dénombrable d'ensembles de chaînes denses et soit $\sigma \in 2^{<\mathbb{N}}$. Il existe un ensemble $\vec{D}$-générique qui étend $\sigma$.*

Preuve. Soit $\sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$ la suite infinie strictement croissante de chaînes définies inductivement comme suit. Initialement, $\sigma_0 = \sigma$. Si $\sigma_n$ est défini, $\sigma_{n+1}$ est une extension stricte de $\sigma_n$ dans $D_n$. Une telle extension existe par densité de $D_n$. Soit $\{G\} = \bigcap_n [\sigma_n]$ ; alors, $G$ est $\vec{D}$-générique et étend $\sigma$. ∎

Nous verrons la contrepartie topologique du théorème précédent avec le lemme 2.14. Le théorème 2.4 dit entre autre que si $(D_n)_{n\in\mathbb{N}}$ est une suite d'ensembles denses correspondants aux contrats $(\mathcal{R}_n)_{n\in\mathbb{N}}$, alors il existe des ensembles $\vec{D}$-génériques, qui satisfont donc tous les contrats simultanément.

Il existe une quantité indénombrable d'ensembles de chaînes denses, et un ensemble $G \in 2^{\mathbb{N}}$ ne peut pas être générique pour tous ces ensembles simultanément, simplement parce que l'ensemble $\{\sigma \in 2^{<\mathbb{N}} : \sigma \not\prec G\}$ est un ensemble dense que $G$ ne rencontre pas. La notion de généricité est donc dépendante d'une collection $\vec{D}$ dénombrable d'ensembles de chaînes denses.

**Suffisamment générique**

Il est courant d'énoncer des résultats de la forme « tout ensemble *suffisamment générique* satisfait telle propriété ». Cela signifie qu'il existe une suite dénombrable $\vec{D} = (D_n)_{n\in\mathbb{N}}$ d'ensembles de chaînes denses telle que tout ensemble $\vec{D}$-générique satisfait la propriété. Notons qu'étant donné n'importe quelle autre suite dénombrable d'ensembles de chaînes denses $\vec{E} = (E_n)_{n\in\mathbb{N}}$, la suite $\{\vec{D}, \vec{E}\}$ est toujours une suite dénombrable d'ensembles de chaînes denses et l'on pourra donc produire un ensemble $\{\vec{D}, \vec{E}\}$-générique. C'est cela qui justifie l'appellation de *suffisamment générique*.

Nous allons reformuler la preuve de la proposition 4-8.2 en termes de densité et de généricité. Nous avons besoin pour cela d'étendre la relation de forcing aux contrats $\Sigma^0_2$, ce qui ne présente pas de difficulté : une chaîne $\sigma$ *force* un tel contrat si ce dernier est satisfait pour tout $G \in [\sigma]$. Nous verrons dans la section 4 que cette idée ne fonctionne plus pour les contrats $\Pi^0_2$.

**Proposition (4-8.2).** Pour tout ensemble non calculable $A$, il existe un ensemble $B$ tel que $B \not\leqslant_T A$ et $A \not\leqslant_T B$. ⋆

Preuve. Soit $A$ un ensemble non calculable. Nous voulons construire un ensemble $B$ satisfaisant les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{R}_e : \exists x \Phi_e^A(x)\uparrow \vee \exists x \Phi_e^A(x)\downarrow \neq B(x) \qquad \mathcal{S}_e : \exists x \Phi_e^B(x)\uparrow \vee \exists x \Phi_e^B(x)\downarrow \neq A(x).$$

Soient $R_e \subseteq 2^{<\mathbb{N}}$ et $S_e \subseteq 2^{<\mathbb{N}}$ l'ensemble des chaînes forçant respectivement $\mathcal{R}_e$ et $\mathcal{S}_e$.

**Densité de l'ensemble** $R_e$. Soit $\sigma \in 2^{<\mathbb{N}}$, et soit $x = |\sigma|$. Deux cas se présentent.

- ▷ Cas 1. On a $\Phi_e^A(x)\downarrow = i$ pour $i \in \{0, 1\}$. Il suffit alors de définir $\tau$ comme l'unique chaîne de longueur $|\sigma| + 1$ étendant $\sigma$ telle que $\tau(x) = 1 - i$. Pour tout $X \in [\tau]$, $X(x) = 1 - i \neq \Phi_e^A(x)$, donc $\tau \in R_e$.
- ▷ Cas 2. On a $\Phi_e^A(x)\uparrow$. Dans ce cas, $R_e = 2^{<\mathbb{N}}$, et $\sigma \in R_e$.

**Densité de l'ensemble** $S_e$. Soit $\sigma \in 2^{<\mathbb{N}}$. Trois cas se présentent.

- ▷ Cas 1. Il existe une entrée $x$ et un ensemble $X \succeq \sigma$ tels que

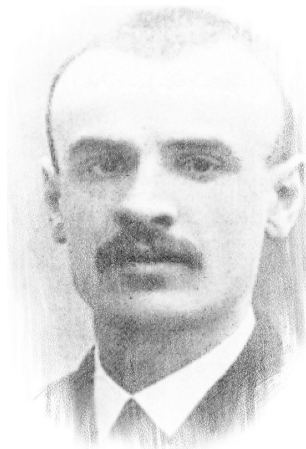
$$\Phi_e^X(x)\downarrow \neq A(x).$$

Dans ce cas, par la propriété de l'usage, il existe une chaîne finie $\tau \succeq \sigma$ telle que $\Phi_e^X(x)\downarrow \neq A(x)$ pour tout $X \in [\tau]$. La chaîne $\tau$ force donc le contrat $\mathcal{S}_e$, donc $\tau \in S_e$.

- ▷ Cas 2. Il existe une entrée $x$ telle que pour tout ensemble $X \succeq \sigma$, on a $\Phi_e^X(x)\uparrow$. Dans ce cas, la chaîne $\sigma$ force déjà le contrat $\mathcal{S}_e$ en s'assurant que $\Phi_e^B(x)\uparrow$. Donc, $\sigma \in S_e$.
- ▷ Cas 3. Aucun des deux cas précédents n'apparaît. Nous avons alors montré dans la preuve initiale de la proposition 4-8.2 que ce cas ne pouvait pas arriver, car l'ensemble $A$ serait calculable, contrairement à notre hypothèse.

Soit $B$ un ensemble $(R_e, S_e)_{e\in\mathbb{N}}$-générique. Un tel ensemble existe par le théorème 2.4. En particulier, $B$ satisfait tous les contrats $\mathcal{R}_e$ et $\mathcal{S}_e$ simultanément, donc $B \not\leqslant_T A$ et $A \not\leqslant_T B$. Cela conclut la preuve de la proposition 4-8.2. ∎

### 2.2. L'approche topologique



René Baire, 1874–1932

La genèse de la méthode des extensions finies, de la généricité, et plus généralement du forcing, se trouve dans les travaux de René Baire, au début du XX$^{\text{e}}$ siècle. Une des motivations de Baire est à l'époque de comprendre un peu mieux certaines fonctions « bizarres », mais qui surviennent naturellement en analyse. Si nous revenons un peu en arrière, dans la première moitié du XIX$^{\text{e}}$ siècle émerge la prise de conscience, notamment aux travers des travaux de Cauchy et Bolzano, qu'une suite $(f_n : \mathbb{R} \to \mathbb{R})_{n\in\mathbb{N}}$ de fonctions continues qui converge point par point n'a pas nécessairement pour limite une fonction continue, et pour cause, une fois les notions de calculabilité correctement étendues pour travailler dans $\mathbb{R}$, les limites de fonctions *effectivement continues* — c'est-à-dire calculables — sont exactement les fonctions $\Delta_2^0$, c'est-à-dire celles pour lesquelles $f(r)$ est calculable uniformément en $r'$, le saut de $r$. Il s'agit d'une simple application du lemme de Schoenfield.

Quelques décennies plus tard, Baire se penche sur ce phénomène, et essaye de mieux comprendre ces fonctions que sont les limites de fonctions continues, et dont les irrégularités les rendent difficile à manipuler et même

à appréhender avec clarté. Il présentera ses résultats dans le cours Peccot « leçons sur les fonctions discontinues », où il développe son fameux outillage mathématique des *catégories de Baire* pour montrer notamment qu'une fonction limite de fonctions continues, si elle n'est plus forcément continue, le sera malgré tout sur un « grand ensemble de points ». Selon la terminologie moderne, les points de discontinuité d'une telle fonction seront une classe *maigre* ou encore de *catégorie 1*. Le théorème 3.20 à venir peut être vu comme une version effective de ce résultat.

Nous avons défini dans la section 8-2 la notion d'ouvert de l'espace de Cantor, comme étant une classe de la forme $\bigcup_{\sigma\in U}[\sigma]$ pour un ensemble $U \subseteq 2^{<\mathbb{N}}$ quelconque. La densité d'un ensemble de chaînes se traduit par une notion de densité de l'ouvert correspondant dans l'espace de Cantor.

**Définition 2.7.** Une classe $\mathcal{B} \subseteq 2^{\mathbb{N}}$ est dite *dense* si elle intersecte tout cylindre $[\sigma]$, c'est-à-dire $[\sigma] \cap \mathcal{B} \neq \emptyset$ pour tout $\sigma \in 2^{<\mathbb{N}}$. ♢

L'exercice suivant lie la notion de densité sur les ouverts de l'espace de Cantor, et sur l'ordre partiel des chaînes binaires.

**Exercice 2.8.** Étant donné $U \subseteq 2^{<\mathbb{N}}$, on note $U^{\prec}$ la clôture par suffixe de $U$, c'est-à-dire $U^{\prec} = \{\tau \in 2^{<\mathbb{N}} \ : \ \exists \sigma \in U \ \tau \succeq \sigma\}$.

Soit $U \subseteq 2^{<\mathbb{N}}$. Montrer que $U^{\prec}$ est dense dans $2^{<\mathbb{N}}$ ssi l'ouvert $\bigcup_{\sigma\in U}[\sigma]$ est dense dans l'espace de Cantor. ◇

Notre but est de définir une notion de classe « négligeable » ou *maigre* pour donner une définition topologique du forcing. On part de l'intuition qu'un cylindre $[\sigma]$ n'est pas maigre. La notion de classe négligeable devrait être close par sous-classe. Ainsi, si une classe contient un ouvert non vide de l'espace de Cantor, elle n'est pas négligeable. On introduit pour formaliser cela la notion d'intérieur.

**Définition 2.9.** On appelle *intérieur* d'une classe $\mathcal{F} \subseteq 2^{\mathbb{N}}$ — que l'on note $\mathrm{int}(\mathcal{F})$ — le plus grand ouvert inclus dans $\mathcal{F}$, c'est-à-dire la réunion de tous les cylindres $[\sigma]$ tels que $[\sigma] \subseteq \mathcal{F}$. ♢

Une classe négligeable doit donc en particulier posséder un intérieur vide.

**Exemple 2.10.** La classe $\{X \in 2^{\mathbb{N}} \ : \ \forall n \ X(2n) = 0\}$ est un fermé d'intérieur vide : il est en effet clair qu'elle ne contient aucun cylindre $[\sigma]$, car il existe toujours des $X \in [\sigma]$ tels que $X(2n) = 1$ pour $n$ suffisamment grand. Son complémentaire $\{X \in 2^{\mathbb{N}} \ : \ \exists n \ X(2n) \neq 0\}$ est donc un ouvert dense, décrit par la réunion des cylindres $[\sigma 1]$ pour toute chaîne $\sigma$ de taille paire.

Nous avons maintenant les éléments en main pour définir la notion de classe maigre.

**Définition 2.11.** Une classe $\mathcal{B} \subseteq 2^{\mathbb{N}}$ est dite *maigre* si $\mathcal{B}$ est incluse dans une réunion dénombrable de classes fermées d'intérieur vide. Le complémentaire d'une classe maigre est dite *co-maigre*. ♢

Notons que, par passage au complémentaire, une classe est co-maigre si elle contient une intersection dénombrable d'ouverts denses. On laisse au lecteur le soin de montrer dans les deux exercices suivants que pour une suite d'ensembles $\vec{W} = (W_n)_{n\in\mathbb{N}}$ de chaînes denses, la classe des ensembles $\vec{W}$-génériques est une classe co-maigre.

**Exercice 2.12.** Soit $\vec{W} = (W_n)_{n\in\mathbb{N}}$ une suite d'ensembles de chaînes telles que chaque $W_n^{\prec}$ est dense (en reprenant la notation de l'exercice 2.8).
Montrer que la classe $\bigcap_n [W_n]$ est co-maigre, où $[W_n] = \bigcup_{\sigma\in W_n} [\sigma]$. ◇

**Exercice 2.13.** Soit $\vec{W} = (W_n)_{n\in\mathbb{N}}$ une suite d'ensembles de chaînes denses. Montrer que la classe $\bigcap_n [W_n]$ contient exactement les ensembles $\vec{W}$-génériques. ◇

Les deux exercices précédents établissent donc un lien entre l'approche du forcing par la méthode des extensions finies et l'approche topologique : si $(\mathcal{R}_n)_{n\in\mathbb{N}}$ est une suite de contrats tels que les ensembles de chaînes correspondants $(W_n)_{n\in\mathbb{N}}$ sont denses, alors la classe des ensembles $\vec{W}$-génériques —qui satisfont tous les contrats simultanément— est co-maigre.

**Digression**

Historiquement, Baire appelle *classes de catégorie 1* les classes maigres, et *classes de catégorie 2* celles qui ne le sont pas. Cela donnera le nom de « théorie des catégories de Baire » à l'étude de ces notions. Notons qu'une classe n'est pas forcément maigre ou co-maigre. En particulier, les classes de catégorie 2 ne sont pas nécessairement co-maigres. En ce qui nous concerne, ce sont réellement les notions de maigre et co-maigre qui nous intéressent, et c'est donc ce vocabulaire que nous utiliserons.

**Fait**

D'après la définition 2.11, il est clair qu'une réunion dénombrable de classes maigres est maigre, et qu'une intersection dénombrable de classes co-maigres est co-maigre.

Une intuition que nous allons étayer dans les développements à venir est que les classes maigres sont « petites » et les classes co-maigres sont « grosses ». Il ne faut pas prendre l'attribution de ces adjectifs comme étant absolue. Il y a d'autres manières de juger de la taille des classes, qui ne coïncident en rien avec le fait que les maigres soit petites et les co-maigres grosses. On peut par exemple trouver des classes maigres de mesure 1 et des classes co-maigre de mesure 0 (voir la partie II).

Ce qu'il faut comprendre plutôt, c'est que les classes co-maigres sont suffisamment grosses pour toujours être stables par intersection dénombrable, et dans le même temps toujours denses, et même dans un sens fort : si $\mathcal{B}$ est une classe co-maigre, alors $\mathcal{B} \cap [\sigma]$ est indénombrable pour tout cylindre $[\sigma]$. Il s'agit en fait d'un renforcement du théorème 2.4. Pour s'en convaincre, se souvenir du fait suivant démontré avec la proposition 8-2.3 et la proposition 8-3.2.

**Fait**

L'intersection d'un nombre fini d'ouverts est un ouvert. Par conséquent, on peut toujours supposer qu'une intersection dénombrable d'ouverts $\bigcap_n \mathcal{U}_n$ est décroissante. Par passage au complémentaire, on peut toujours supposer qu'une réunion dénombrable de fermés est croissante.

Rappelons qu'une classe $\mathcal{F} \subseteq 2^{\mathbb{N}}$ est *parfaite* si elle est l'image d'une injection continue de $2^{\mathbb{N}}$ vers $2^{\mathbb{N}}$ (voir la section 8-2.4), c'est-à-dire que $\mathcal{F}$ est de la forme $[T]$ pour un f-arbre $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ (voir la section 7-5). Le lemme suivant montre qu'une classe co-maigre a la puissance du continu.

**Lemme 2.14.** Une classe co-maigre de l'espace de Cantor contient une sous-classe parfaite de points dans chaque cylindre $[\sigma]$. ⋆

Preuve. Soit $\mathcal{B}$ une classe co-maigre, et soit $\bigcap_n \mathcal{U}_n \subseteq \mathcal{B}$ une intersection d'ouverts denses. On peut supposer sans perte de généralité que $\mathcal{U}_{n+1} \subseteq \mathcal{U}_n$. Le lecteur peut s'aider de la figure 2.2 pour une représentation graphique de la construction qui suit.

Soit $\sigma \in 2^{<\mathbb{N}}$. Comme l'ouvert $\mathcal{U}_0$ est dense, $\mathcal{U}_0 \cap [\sigma 0]$ est non vide, et il y a donc une chaîne $\sigma_0 \succ \sigma 0$ telle que $[\sigma_0] \subseteq \mathcal{U}_0$. De la même manière il y a une chaîne $\sigma_1 \succ \sigma 1$ telle que $[\sigma_1] \subseteq \mathcal{U}_0$. Supposons que pour toute chaîne $\tau$ de taille $n+1$ on ait défini des chaînes $\sigma_\tau \succeq \sigma$ deux à deux incomparables et telles que $[\sigma_\tau] \subseteq \mathcal{U}_n$. Pour chacune de ces chaînes $\tau$ et pour chaque $i \in \{0,1\}$, on définit $\sigma_{\tau i}$ comme étant une chaîne qui étend $\sigma_\tau i$ et telle que $[\sigma_{\tau i}] \subseteq \mathcal{U}_{n+1}$, ce qui est possible car $\mathcal{U}_{n+1}$ est dense.

Il est clair que pour tout ensemble $X$ la classe $\bigcap_{\tau \prec X} [\sigma_\tau] \subseteq \bigcap_n \mathcal{U}_n$ contient un unique élément $Y_X \in [\sigma] \cap \bigcap_n \mathcal{U}_n$. On vérifie sans peine que la fonction $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ définie par $T(\tau) = \sigma_\tau$ est un f-arbre dont les chemins sont les éléments $Y_X = T(\sigma_0) \prec T(\sigma_1) \prec \dots$ pour $X = \sigma_0 \prec \sigma_1 \prec \dots$

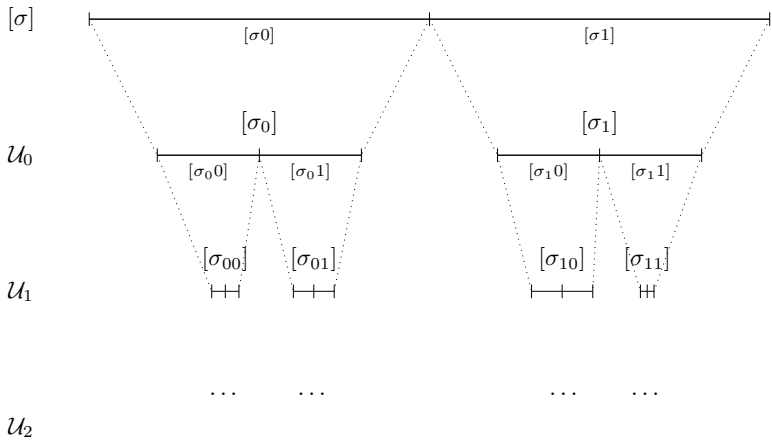



FIGURE 2.15 – *Illustration de la construction d'une sous-classe parfaite de points dans* $[\sigma] \cap \bigcap_n \mathcal{U}_n$. *Comme chaque ouvert* $\mathcal{U}_n$ *est dense, on peut trouver pour toute chaîne* $\sigma_\tau i$ *une extension* $\sigma_{\tau i} \succeq \sigma_\tau i$ *telle que* $[\sigma_{\tau i}] \subseteq \mathcal{U}_n$.

Ainsi, $\mathcal{B}$ contient une sous-classe parfaite de points dans le cylindre $[\sigma]$, ce qui achève la démonstration du lemme 2.14. ∎

Notons en particulier comme conséquence du lemme précédent qu'une réunion dénombrable de fermés d'intérieur vide est d'intérieur vide : si une telle réunion de fermés contenait un cylindre $[\sigma]$, son complémentaire — une classe co-maigre — ne pourrait contenir de point dans $[\sigma]$, ce qui serait une contradiction.

Nous voyons à présent l'équivalent topologique de la définition 2.1 de forcing pour les contrats $\Sigma^0_1$ et $\Pi^0_1$.

**Définition 2.16.** Soit $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$. Soit $\mathcal{B}_\mathcal{R}$ la classe des éléments qui satisfont $\mathcal{R}$. Alors, $\sigma$ force $\mathcal{R}$ ssi $[\sigma] \subseteq \mathcal{B}_\mathcal{R}$. Notons que si $\mathcal{R}$ est $\Sigma^0_1$, alors $\mathcal{B}_\mathcal{R}$ est un ouvert effectif, et que si $\mathcal{R}$ est $\Pi^0_1$, alors $\mathcal{B}_\mathcal{R}$ est un fermé effectif. ♢

Notons par exemple que si la classe des éléments satisfaisant un contrat $\Pi^0_1$ est d'intérieur vide, alors aucune chaîne ne force ce contrat : la classe des éléments ne satisfaisant pas le contrat est un ouvert dense, et contiendra tout élément suffisamment générique. Cela nous amène à l'étude des ensembles qui sont dans « suffisamment d'ouverts denses » : les 1-génériques et les faiblement 1-génériques, que nous voyons à présent.

# 3. Généricité effective

Jockusch fut sans doute l'un des premiers à comprendre l'utilité que pouvaient avoir les idées de Cohen en calculabilité, et il initia l'étude d'une version effective des concepts de Cohen, avec les notions de 1-généricité et 1-généricité faible, résultant de l'application du forcing à tous les contrats $\Sigma^0_1$ et $\Pi^0_1$.

La généricité peut être vue à la fois comme une notion de force et de faiblesse. Un ensemble suffisamment générique sera par exemple toujours de degré hyperimmune. En revanche, nous verrons que les ensembles suffisamment génériques ne peuvent pas calculer l'arrêt, ni même de fonction DNC. De manière générale, la méthode des extensions finies satisfait des propriétés de force et de faiblesse de la même manière : en prouvant la densité de certains ensembles bien choisis.

## 3.1. Ensembles faiblement 1-génériques

Nous commençons par présenter les ensembles faiblement 1-génériques, introduits par Kurtz durant sa thèse de doctorat, effectuée sous la direction de Jockusch.

**Définition 3.1 (Kurtz [126]).** Un ensemble $G \in 2^{\mathbb{N}}$ est *faiblement 1-générique* s'il est générique pour tous les ensembles c. e. denses. Autrement dit, $G$ est faiblement 1-générique si $G$ appartient à toutes les classes $\Sigma^0_1$ denses de l'espace de Cantor. ♢

Notons qu'il n'existe qu'une quantité dénombrable de classes $\Sigma^0_1$ denses. La notion d'ensemble faiblement 1-générique s'avère n'être pas assez restrictive pour receler des propriétés normalement inhérentes aux génériques, mais en termes de degré Turing, la notion présente un certain intérêt, notamment via la caractérisation suivante.

**Théorème 3.2 (Kurtz [126])**
*Soit $G \subseteq \mathbb{N}$. Les énoncés suivants sont équivalents.*

*(1) $G$ est de degré hyperimmune.*

*(2) $G$ calcule une fonction qui est égale infiniment souvent à toute fonction calculable.*

*(3) $G$ calcule un ensemble faiblement 1-générique.*

Preuve. Montrons d'abord (1) → (3), l'implication la plus difficile.
Soit $f \leqslant_T G$ une fonction qui n'est bornée par aucune fonction calculable. On suppose sans perte de généralité que $f$ est croissante. Notons que pour toute fonction calculable $g$, il y a une infinité de valeurs $n$ telles

que $f(n) > g(n)$. On calcule à partir de la fonction $f$ un ensemble $G \in 2^{\mathbb{N}}$ qui appartient à toute classe $\Sigma_1^0$ dense. Soit $(W_e)_{e\in\mathbb{N}}$ une énumération des sous-ensembles $\Sigma_1^0$ de $2^{<\mathbb{N}}$. On construit $G$ par approximations successives $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$

On décrit d'abord une procédure récursive à effectuer à chaque fois que l'on veut concaténer une chaîne $\tau$ à une chaîne $\sigma$ que l'on a jusqu'à présent calculé. Cette procédure, que nous nommerons $R$, prend un troisième paramètre : un entier $e$ qui correspond au plus petit entier tel que $\sigma\tau$ est énuméré dans $W_e$ à l'étape de calcul $f(|\sigma|)$. On notera $R(\sigma, \tau, e)$ pour le résultat de l'appel à cette procédure. Notons enfin que certains entiers sont marqués comme « satisfaits » au moment où la procédure est appelée : ce sont les entiers $e$ tels que $\sigma$ étend une chaîne de $W_e$ à l'étape de calcul courante.

La procédure $R(\sigma, \tau, e)$ fait la chose suivante : pour chaque préfixe $\tau' \preceq \tau$ dans l'ordre, elle cherche le plus petit entier $e' < e$ qui n'est pas satisfait et tel qu'une chaîne de la forme $\sigma\tau'\rho$ soit énumérée dans $W_{e'}[f(|\sigma\tau'|)]$. Si un tel entier est trouvé, la procédure renvoie alors le résultat de l'appel récursif à $R(\sigma\tau', \rho, e')$. Sinon, elle renvoie $\sigma\tau$. Notons que la diminution de la valeur du dernier paramètre dans les appels récursifs fait que la procédure s'arrête nécessairement.

À l'étape 0, on définit $\sigma_0 = \epsilon$. Supposons $\sigma_t$ défini à l'étape $t$. À l'étape $t+1$, on cherche le plus petit entier $e \leqslant t+1$ non satisfait tel qu'une chaîne de la forme $\sigma_t\tau$ soit énumérée dans $W_e[f(|\sigma_t|)]$. Si l'on trouve un tel entier $e$, on définit $\sigma_{t+1}$ comme étant $R(\sigma_t, \tau, e)$. Sinon, $\sigma_{t+1}$ comme étant $\sigma 0$. Cela conclut la construction.

Notons que si $W_e$ décrit un ouvert dense, alors la fonction $f_e$ qui à $n$ associe le plus petit temps de calcul $t$ tel que toutes les chaînes de taille $n$ ont une extension dans $W_e[t]$ est une fonction calculable et totale. On a en particulier $f_e(n) < f(n)$ pour une infinité de valeurs $n$. Supposons que $W_e$ décrit un ouvert dense, que $e$ n'est pas satisfait au temps $t$ et que tous les $e' < e$ qui sont satisfait à un moment de la construction sont satisfaits au temps $t$. Soit $n$ le plus petit entier supérieur ou égal à $|\sigma_t|$ tel que $f(n) > f_e(n)$. Soit $s \geqslant t$ le plus petit entier tel que $|\sigma_s| \leqslant n < |\sigma_{s+1}|$. Si $|\sigma_s| = n$, alors par minimalité de $e$ l'algorithme définit $\sigma_{s+1} = \sigma_s\tau$ avec $\sigma_s\tau \in W_e[f(n)]$. Sinon alors, par construction, au moment de définir $\sigma_{s+1} = \sigma_s\tau$ pour une certaine chaîne $\tau$, l'algorithme vérifie pour tout préfixe $\tau' \preceq \tau$, que l'on n'a pas une extension de $\sigma_s\tau'$ énumérée dans $W_e[f(|\sigma_s\tau'|)]$, et en particulier pour le préfixe $\tau'$ tel que $|\sigma_s\tau'| = n$. Si c'est le cas, l'algorithme est relancé sur cette extension. Comme c'est effectivement le cas par hypothèse, et par minimalité de $e$, on aura en fait $\sigma_s\tau \in W_e[f(n)]$ pour $\sigma_{s+1} = \sigma_s\tau$. On en conclut que l'ensemble $G = \sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$ appartient à tous les ouverts denses.

Montrons l'implication (3) → (2).
Soit $G$ un ensemble faiblement 1-générique. Notons que si $f$ est une fonction totale calculable alors l'ensemble $W_f = \{\sigma 0^{f(|\sigma|)}1 : \sigma \in 2^{<\mathbb{N}}\}$ décrit une classe $\Sigma^0_1$ dense. On calcule à partir de $G$ la fonction $g$ qui à $n$ associe le nombre maximal $g(n)$ de 0 tel que $G \upharpoonright_n 0^{g(n)} \prec G$. Comme pour toute fonction totale calculable $f$ l'ensemble $G$ appartient à $W_f$, il est clair que $g$ est égale au moins une fois à toute fonction calculable. Si $g$ n'était égale qu'un nombre fini de fois à une fonction calculable donnée, on pourrait modifier un nombre fini de valeurs de cette fonction pour avoir une fonction calculable qui n'est jamais égale à $g$. Donc, $g$ est égale infiniment souvent à toute fonction calculable.

Montrons enfin (2) → (1).
Soit $f$ une fonction égale infiniment souvent à toute fonction calculable. Alors, $f+1$ est infiniment souvent au-dessus de toute fonction calculable.■

On peut relativiser la notion de 1-généricité faible à n'importe quel oracle.

**Définition 3.3 (Kurtz [126]).** Soit $A \in 2^{\mathbb{N}}$. Un ensemble $G$ est *faiblement 1-générique relativement à* $A$ si $G$ rencontre $W$ pour tout ensemble de chaînes $\Sigma^0_1(A)$ dense $W$. ♢

La hiérarchie des sauts Turing permet de définir une hiérarchie de généricité comme suit.

**Définition 3.4 (Kurtz [126]).** Un ensemble $G \in 2^{\mathbb{N}}$ est *faiblement $n$-générique* s'il est faiblement 1-générique relativement à $\emptyset^{(n-1)}$, c'est-à-dire s'il rencontre tous les ensembles de chaînes $\Sigma^0_1(\emptyset^{(n-1)})$ denses, ou encore tous les ensembles de chaînes $\Sigma^0_n$ denses. ♢

Certaines implications du théorème 3.2 se généralisent à tout oracle $A$.

**Exercice 3.5.** Montrer que tout ensemble $G$ faiblement 1-générique relativement à $A$ calcule une fonction $g : \mathbb{N} \to \mathbb{N}$ qui est égale infiniment souvent à toute fonction $A$-calculable. ◇

**Exercice 3.6.** Soit $n \geqslant 1$, et soit $G \in 2^{\mathbb{N}}$. Montrer que si $G$ calcule une fonction qui est égale infiniment souvent à toutes les fonctions $A$-calculables, alors $G$ calcule une fonction $A$-hyperimmune. ◇

La direction (1) → (3) du théorème 3.2 ne se généralise pas en règle générale. Dans la hiérarchie des sauts Turing, cela fonctionne uniquement à la première étape.

**Proposition 3.7 (Andrews, Gerdes et Miller [7]).**
Toute fonction hyperimmune relativement à $\emptyset'$ calcule un ensemble faiblement 2-générique. ⋆

L'idée pour montrer la proposition précédente est d'utiliser le fait que les objets $\emptyset'$-calculables sont calculables à la limite. En revanche, à partir de $n \geqslant 3$, les ensembles faiblement $n$-génériques ne peuvent plus être construits simplement à l'aide de fonctions échappant à des collections de fonctions, au sens suivant.

**Définition 3.8.** Soit $\mathcal{F}$ une collection de fonctions $\mathbb{N} \to \mathbb{N}$. Une fonction $f$ est *$\mathcal{F}$-échappante* si pour tout $g \in \mathcal{F}$, il existe un entier $n$ tel que $f(n) > g(n)$. ♢

Le théorème suivant exprime une profonde différence structurelle entre les fonctions échappantes et les degrés génériques, au sens où quel que soit un oracle $A$, il existe une fonction $A$-hyperimmune qui ne calcule pas d'ensemble faiblement 3-générique.

**Théorème 3.9 (Andrews, Gerdes et Miller [7])**
*Pour toute collection dénombrable de fonctions $\mathcal{F}$, il existe une fonction $\mathcal{F}$-échappante qui n'est pas de degré faiblement 3-générique.*

Preuve. Nous utiliserons une variante de la notion de f-arbre définie dans la section 7-5. Nous allons définir une fonction $\Delta^0_3$ totale $T : \mathbb{N}^{<\mathbb{N}} \to \mathbb{N}^{<\mathbb{N}}$ telle que :

(1) $\operatorname{dom} T$ (le domaine de $T$) est un ensemble clos par préfixe ;

(2) pour tous $\sigma, \tau \in \operatorname{dom} T$, $\sigma \preceq \tau$ si et seulement si $T(\sigma) \preceq T(\tau)$ ;

(3) pour tout $\sigma \in \mathbb{N}^{<\mathbb{N}}$ et $n \in \mathbb{N}$, il existe un $m \geqslant n$ tel que $T_s(\sigma n)$ étend $T_s(\sigma)m$.

La fonction $T$ s'étend en une fonction de $\mathbb{N}^{\mathbb{N}}$ vers $\mathbb{N}^{\mathbb{N}}$, en définissant pour tout $X \in \mathbb{N}^{\mathbb{N}}$, $T(X)$ comme l'unique élément de la classe $\bigcap_{\sigma \prec X} [T(\sigma)]$. Un *chemin* de $T$ est une suite $P \in \mathbb{N}^{\mathbb{N}}$ dont une infinité de segments initiaux appartiennent à $\operatorname{Im} T$. Autrement dit, un chemin est une suite $P \in 2^{\mathbb{N}}$ de la forme $P = T(X)$ pour un $X \in \mathbb{N}^{\mathbb{N}}$. On notera $[T]$ l'ensemble des chemins de $T$. Par (3), pour toute collection dénombrable de fonctions $\mathcal{F}$, il existe un chemin $f \in [T]$ qui est $\mathcal{F}$-échappant. Nous allons construire $T$ de telle sorte que pour tout chemin $f \in [T]$, $f$ n'est pas de degré faiblement 3-générique.

Le lecteur peut s'aider de la figure 3.10 pour comprendre la construction qui suit. Soit $(\sigma_s)_{s \in \mathbb{N}}$ une énumération calculable de $\mathbb{N}^{<\mathbb{N}}$ qui ne fait apparaître chaque chaîne qu'après avoir énuméré ses préfixes. En particulier, $\sigma_0 = \epsilon$. Au début de l'étape $s$, nous aurons déjà défini $T$ sur $\sigma_0, \dots, \sigma_s$.

Nous supposerons également défini pour chaque $t \leqslant s$, un réservoir c. e. infini $V_{t,s} \subseteq \mathbb{N}^{<\mathbb{N}}$ de chaînes étendant $T(\sigma_s)$. Nous allons nous assurer que pour tout $n \in \mathbb{N}$, $T(\sigma_s n)$ étend une chaîne de $V_{t,s}$. Simultanément, nous allons créer pour chaque $e$ un ensemble $\Sigma^0_1(\emptyset'')$ dense $U_e \subseteq 2^{<\mathbb{N}}$ tel que si $\Phi_e^P$ est totale pour un chemin $P \in [T]$, alors $P$ ne rencontre pas $U_e$. Ainsi, quel que soit le chemin $P \in [T]$, $\Phi_e^P$ ne sera pas un ensemble faiblement 3-générique.

Initialement, $f(\epsilon) = \epsilon$ et $V_{0,0} = \mathbb{N}$. À l'étape $s = \langle e, i\rangle$, nous allons nous assurer que toute chaîne de longueur $i$ a une extension dans $U_e$. L'ensemble $\{\sigma_t : t \leqslant s\}$ forme un arbre fini, et pour chaque feuille $\tau$ de cet arbre, $\Phi_e^{T(\tau)}$ peut avoir des valeurs différentes. L'ensemble $U_e$ doit donc ajouter des extensions à toutes les chaînes de longueur $i$, tout en s'assurant qu'elle évitera $\Phi_e^{T(\sigma_t)}$ pour tout $t \leqslant s$. Nous fixons donc une longueur suffisamment grande, $k = i+s+1$, de telle sorte que si l'on construit un ensemble de chaînes binaires « interdites » $\rho_0, \ldots, \rho_s$ chacune de longueur $k$, toute chaîne binaire de longueur $i$ admet une extension de longueur $k$ évitant les chaînes interdites.

Pour chaque $t \leqslant s$, on demande à $\emptyset''$ s'il existe une chaîne $\rho_s$ de taille $k$ telle qu'il existe une infinité de chaînes binaires $\mu \in V_{t,s}$ ayant une extension $\mu' \succeq \mu$ pour laquelle $\Phi_e^{\mu'} \restriction_k = \rho_s$. Si c'est le cas, on définit $V_{t,s+1}$ comme l'ensemble de ces $\mu'$. Notons que $V_{t,s+1}$ est alors encore calculatoirement énumérable. Sinon, pour toute chaîne $\rho$ de taille $k$, seul un nombre fini de chaînes de $V_{t,s}$ possède une extension qui sera envoyée vers une extension de $\rho$ via $\Phi_e$. En particulier, on peut enlever un nombre fini de chaînes de $V_{t,s}$ de manière à ce qu'aucune extension des chaînes restantes ne sera jamais envoyé vers une chaîne de taille plus grande que $k$. On prend alors une chaîne $\rho_s$ arbitraire, et l'on définit $V_{t,s+1}$ comme la restriction de $V_{t,s}$ aux chaînes $\mu$ qui n'ont pas d'extension $\mu'$ envoyée vers une chaîne de taille plus grande que $k$ via $\Phi$. Comme nous retirons un nombre fini de chaînes, $V_{s,t+1}$ est encore c. e.

Nous nous retrouvons donc avec un ensemble de chaînes binaires $\rho_0, \ldots, \rho_s$, chacune de longueur $k$, de telle sorte que pour tout chemin $P \in [T]$, si $\Phi_e^P$ est total, alors $\Phi_e^P \restriction_k = \rho_t$ pour un $t \leqslant s$. On énumère dans $U_e$ toutes les chaînes de longueur $k$ autres que $\rho_0, \ldots, \rho_s$. En excluant ces chaînes, on s'assure que pour tout chemin $P \in [T]$, si $\Phi_e^P$ est total, alors il ne rencontrera pas $U_e$. La longueur $k$ étant suffisamment grande, toute chaîne de longueur $i$ a une extension dans $U_e$.

Enfin, nous définissons $T(\sigma_{s+1})$. Supposons que $\sigma_{s+1} = \sigma_t n$ pour un $t \leqslant s$ et $n \in \mathbb{N}$. On choisit $\tau \in V_{t,s+1}$, on le retire de $V_{t,s+1}$, et l'on pose $T(\sigma_{s+1}) = \tau$. Enfin, on définit $V_{s+1,s+1} = \{\tau m : m \in \mathbb{N}\}$. Cela termine la preuve du théorème 3.9. ∎

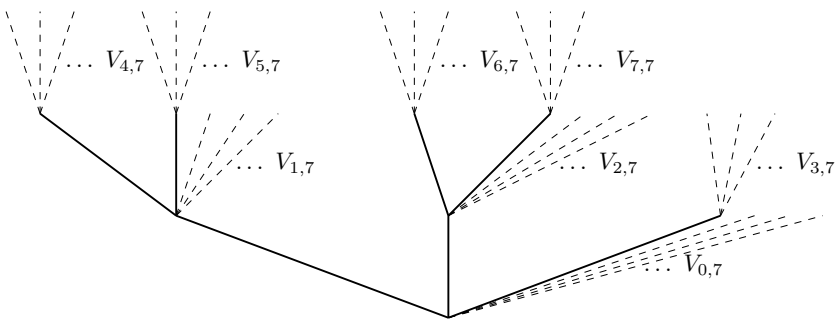



FIGURE 3.10 – *Illustration de la preuve : à l'étape* 8*, on va restreindre chacun des réservoirs de $V_{0,7}$ à $V_{7,7}$ en supprimant certaines de leurs branches, et en étendant d'autres, de manière à ce que la fonctionnelle courante $\Phi_e$ évite, pour chacune des branches des réservoirs $V_{i,8}$, un ouvert dense que l'on est en train d'énumérer à l'aide du double saut. Une fois les réservoirs restreints, on complète notre arbre en prenant une extension dans chacun d'entre eux, ce qui crée pour chaque extension un nouveau réservoir, et ainsi de suite.*

**Exercice 3.11. (⋆⋆)** Modifier la preuve du théorème précédent pour montrer que pour toute collection dénombrable de fonctions $\mathcal{F}$, il existe une fonction $\mathcal{F}$-échappante qui ne calcule aucune fonction qui est égale infiniment souvent à toute fonction $\emptyset''$-calculable. ◇

### 3.2. Ensembles 1-génériques

Nous voyons à présent la 1-généricité, une notion un peu plus forte et beaucoup plus riche que la 1-généricité faible. Les ensembles faiblement 1-génériques sont ceux résultant du forcing pour les contrats $\Sigma^0_1$ denses. Mais qu'en est-il des contrats $\Sigma^0_1$ qui ne sont pas denses ?

La 1-généricité correspond au premier niveau de forcing permettant de s'attaquer aux contrats $\Sigma^0_1/\Pi^0_1$ quelconques, via le théorème suivant, qui sera démontré à la fin de la sous-section qui suit (nous renvoyons le lecteur à la définition 2.1 pour la notation $\Vdash^*$).

**Théorème 3.12**
*Soit $G$ un ensemble* 1*-générique. Soit $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$. Alors, $G$ satisfait $\mathcal{R}$ si, et seulement si, il existe un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \mathcal{R}$.*

### 3.2.1. Nature des ensembles 1-génériques

**Définition 3.13.** Une chaîne $\sigma \in 2^{<\mathbb{N}}$ *évite* un ensemble $W \subseteq 2^{<\mathbb{N}}$ (noté $\sigma \perp W$) si non seulement $\sigma \notin W$, mais également aucune extension de $\sigma$ n'est dans $W$. ♢

**Notation**

On note $W^{\perp} = \{\tau \in 2^{<\mathbb{N}} : \tau \perp W\}$.

**Lemme 3.14.** Soit $W \subseteq 2^{<\mathbb{N}}$ un ensemble arbitraire.
Alors, l'ensemble $W \cup W^{\perp}$ est dense. ⋆

Preuve. Soit $\sigma \in 2^{<\mathbb{N}}$. Soit $\sigma$ possède une extension dans $W$, et donc dans $W \cup W^{\perp}$, soit $\sigma$ évite $W$, auquel cas $\sigma \in W^{\perp}$. ∎

Notons que si $W$ est un ensemble dense, alors $W^{\perp} = \emptyset$. Souvenons-nous que la densité d'un ensemble de chaînes $W$ clos par suffixe sur $2^{<\mathbb{N}}$ correspond à la densité sur $2^{\mathbb{N}}$ de son ouvert correspondant $[W] = \bigcup_{\sigma \in W} [\sigma]$. L'ensemble $W^{\perp}$ correspond également à un ouvert : l'intérieur du complémentaire de l'ensemble $[W]$, c'est-à-dire la réunion de tous les cylindres $[\sigma]$ inclus dans le complémentaire de $[W]$.

Si on le reformule dans l'espace de Cantor, le lemme 3.14 énonce que pour n'importe quelle classe ouverte $\mathcal{U} \subseteq 2^{\mathbb{N}}$, la classe $\mathcal{U} \cup \operatorname{int}(2^{\mathbb{N}} \setminus \mathcal{U})$ est un ouvert dense. Si $\mathcal{U}$ est à la base dense, alors $\operatorname{int}(2^{\mathbb{N}} \setminus \mathcal{U}) = \emptyset$, sinon on « densifie » $\mathcal{U}$ en y ajoutant l'intérieur de son complémentaire. Nous sommes maintenant prêts à définir la notion d'ensemble 1-générique.

**Définition 3.15 (Jockusch [100]).** Un ensemble $G \in 2^{\mathbb{N}}$ est 1-*générique* s'il est $\{W_e \cup W_e^{\perp} : e \in \mathbb{N}\}$-générique, où $W_e$ est l'ensemble de chaînes calculatoirement énumérable de code $e$. De manière équivalente, $G$ est 1-générique si $G \in \mathcal{U} \cup \operatorname{int}(2^{\mathbb{N}} \setminus \mathcal{U})$ pour toute classe $\Sigma^0_1$ $\mathcal{U}$. ♢

Comme signalé plus haut, si $W \subseteq 2^{<\mathbb{N}}$ est un ensemble dense, alors $W^{\perp} = \emptyset$. Il s'ensuit que tout ensemble 1-générique rencontre tout ensemble $\Sigma^0_1$ dense, et est donc faiblement 1-générique. En particulier, les degrés des ensembles faiblement 1-génériques coïncidant avec les degrés hyperimmunes, tout ensemble 1-générique est de degré hyperimmune, et donc non calculable.

Si l'on regarde la contraposée de la notion de 1-généricité, un ensemble $G$ n'est pas 1-générique s'il existe un ensemble c. e. $W \subseteq 2^{<\mathbb{N}}$ ne contenant aucun préfixe de $G$ et tel que $W$ « est dense le long de $G$ », c'est-à-dire que pour tout $\sigma \prec G$ il existe $\tau \succeq \sigma$ avec $\tau \in W$ et $\tau \not\prec G$. Cette idée est illustrée dans la figure 3.16.

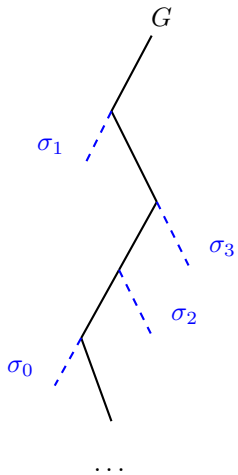



FIGURE 3.16 – *Une illustration d'un ensemble $G$ qui n'est pas 1-générique : on peut énumérer des chaînes $\sigma_0, \sigma_1, \ldots$* densément le long de $G$, *et sans jamais énumérer un préfixe de $G$.*

Nous voyons à présent formellement pourquoi le théorème 3.12 annoncé est vrai.

**Proposition 3.17.** Soit $\mathcal{R}_e$ un contrat $\Sigma^0_1$. Alors, l'ensemble $W \subseteq 2^{<\mathbb{N}}$ des chaînes forçant $\mathcal{R}_e$ est $\Sigma^0_1$. De plus, $W^\perp$ est l'ensemble des chaînes forçant $\neg\mathcal{R}_e$ et il est $\Pi^0_1$. ⋆

PREUVE. Le contrat $\mathcal{R}_e$ étant $\Sigma^0_1$, il peut s'écrire de la forme $\Phi(G,0)\!\downarrow$ pour une fonctionnelle $\Phi$. Une chaîne $\sigma$ force $\mathcal{R}_e$ si $\Phi(X,0)\!\downarrow$ pour tout $X \in [\sigma]$. Par le lemme de König, $\sigma$ force $\mathcal{R}_e$ s'il existe une longueur $n$ telle que pour tout $\tau \succeq \sigma$ de longueur $n$, $\Phi(\tau,0)\!\downarrow$. L'ensemble

$$W = \{\sigma \in 2^{<\mathbb{N}} : \exists n\ \forall \tau \in 2^n\ (\tau \succeq \sigma \to \Phi(\tau,0)\!\downarrow)\}$$

est $\Sigma^0_1$ (où $2^n$ désigne l'ensemble des chaînes de taille $n$).

Montrons que $W^\perp$ est l'ensemble des chaînes $\sigma$ qui forcent $\neg\mathcal{R}_e$. Si $\sigma$ ne force pas $\neg\mathcal{R}_e$, alors il existe un $X \in [\sigma]$ tel que $\Phi(X,0)\!\downarrow$. Par la propriété de l'usage, pour $n$ suffisamment grand et pour tout $Y \in [X\upharpoonright_n]$, $\Phi(Y,0)\!\downarrow$. On peut supposer $n \geqslant |\sigma|$. Soit $\tau = X\upharpoonright_n$. Alors, $\tau \in W$, donc $\sigma \notin W^\perp$. Par contraposée, si $\sigma \in W^\perp$, alors $\sigma$ force $\neg\mathcal{R}_e$. Supposons maintenant que $\sigma \notin W^\perp$. Alors, il existe $\tau \succeq \sigma$ tel que $\tau \in W$. En particulier, il existe un $X \in [\tau] \subseteq [\sigma]$ tel que $\Phi(X,0)\!\downarrow$. Il s'ensuit que $\sigma$ ne force pas $\neg\mathcal{R}_e$.

Comme l'ensemble $W$ est $\Sigma^0_1$, il est clair que l'ensemble $W^\perp$ est $\Pi^0_1$. ∎

**Corollaire 3.18**
*Soit $\mathcal{R}_e$ un contrat $\Sigma^0_1$. Alors, l'ensemble $D$ des chaînes qui forcent $\mathcal{R}_e$ ou qui forcent $\neg\mathcal{R}_e$ est dense.*

PREUVE. Immédiat par le lemme 3.14 et la proposition 3.17. ∎

Notons que tout ensemble satisfait la formule « $\mathcal{R}_e \vee \neg\mathcal{R}_e$ », et donc que toute chaîne force « $\mathcal{R}_e \vee \neg\mathcal{R}_e$ ». En revanche, il est beaucoup plus fort pour une chaîne $\sigma$ de forcer $\mathcal{R}_e$ ou de forcer $\neg\mathcal{R}_e$, car tout ensemble $A \in [\sigma]$ doit avoir le même comportement vis-à-vis de $\mathcal{R}_e$.

On peut à présent montrer le théorème 3.12 annoncé : si $G$ est un ensemble 1-générique et $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$, alors $G$ satisfait $\mathcal{R}$ ssi il existe un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \mathcal{R}$.

Preuve du théorème 3.12. Soit $G$ un ensemble 1-générique. Soit $\mathcal{R}$ un contrat $\Sigma^0_1$. Si un préfixe $\sigma \prec G$ force $\mathcal{R}$ ou $\neg\mathcal{R}$, alors par définition $G$ satisfait respectivement $\mathcal{R}$ ou $\neg\mathcal{R}$.

Réciproquement, supposons que $G$ satisfasse $\mathcal{R}$. Soit $W$ l'ensemble c. e. des chaînes qui forcent $\mathcal{R}$. Comme $G$ est 1-générique, il rencontre $W \cup W^\perp$. Si $G$ rencontre $W^\perp$, alors $\sigma$ force $\neg\mathcal{R}$ pour un préfixe $\sigma \prec G$, et $G$ satisfait donc $\neg\mathcal{R}$, ce qui est une contradiction. Ainsi, $G$ rencontre $W$, et un préfixe $\sigma \prec G$ force donc $\mathcal{R}$.

Symétriquement, si $G$ satisfait $\neg\mathcal{R}$, alors un préfixe $\sigma \prec G$ force $\neg\mathcal{R}$. ∎

### 3.2.2. Propriétés des ensembles 1-génériques

De manière générale, la fonction qui à $X$ associe $X'$ n'est pas continue. En effet, il est parfois nécessaire de connaître une infinité de bits de $X$ pour savoir si $n \in X'$. René Baire a montré — sous une autre forme bien sûr — que cette fonction était en revanche continue sur une classe co-maigre de points. La 1-généricité est le niveau de généricité requis pour rendre compte de ce théorème.

**Définition 3.19.** Un ensemble $G \in 2^{\mathbb{N}}$ est *low généralisé* si

$$G' \leqslant_T G \oplus \emptyset'. \qquad \diamondsuit$$

Si un ensemble $G$ est low généralisé, alors non seulement la fonction qui à $X$ associe $X'$ est continue en $G$, mais plus encore elle est calculable en $G \oplus \emptyset'$. Un ensemble quelconque n'a *a priori* aucune raison d'être low généralisé. À titre d'exemple, $\emptyset'$ ne l'est pas, mais c'est le cas pour les ensembles 1-génériques.

**Théorème 3.20**
*Les ensembles 1-génériques sont low généralisés.*

Preuve. Pour tout entier $e$, on définit la classe $\mathcal{U}_e = \{X \ : \ \Phi_e(X, e)\downarrow\}$. Soit $W_e \subseteq 2^{<\mathbb{N}}$ un ensemble $\Sigma^0_1$ qui représente $\mathcal{U}_e$, c'est-à-dire tel que

$$[W_e] = \mathcal{U}_e, \quad \text{où } [W_e] = \bigcup_{\sigma \in W_e} [\sigma].$$

Notons que $e \in G'$ ssi $G \in \mathcal{U}_e$. On a par ailleurs $G \in \mathcal{U}_e$ ssi il existe $\tau \preceq G$ tel que $\tau \in W_e$, et par définition de la 1-généricité de $G$ on a $G \notin \mathcal{U}_e$ ssi il existe $\tau \preceq G$ tel que $\sigma \notin W_e$ pour tout $\sigma$ comparable avec $\tau$. La question de savoir si $\sigma \notin W_e$ pour tout $\sigma$ comparable avec $\tau$ est $\Pi^0_1$ uniformément en $\tau$, et donc peut être posée à $\emptyset'$. Pour savoir si $e \in G'$, il suffit donc de chercher un préfixe $\tau$ de $G$ tel que l'on est dans un cas ou dans l'autre, ce qui arrivera nécessairement. ∎

Nous avons vu que tout ensemble 1-générique était de degré hyperimmune, et donc non calculable. Il ne s'agit donc pas d'une propriété de faiblesse. Nous allons maintenant voir que ce n'est pas non plus une notion de force, au sens où certaines puissances calculatoires ne pourront jamais être atteintes par les degrés 1-génériques. En particulier, aucun degré 1-générique ne calcule $\emptyset'$.

**Théorème 3.21 (Demuth et Kučera [45])**
*Un ensemble 1-générique n'est pas de degré DNC.*

Preuve. Supposons que $n \mapsto \Phi(G, n)$ soit une fonction DNC. On considère la classe $\mathcal{U} = \{X \ : \ \exists n \ \Phi(X, n) \downarrow = \Phi_n(n) \downarrow\}$. Par hypothèse, $G \notin \mathcal{U}$. Considérons une chaîne $\sigma$. On définit par le théorème de point fixe de Kleene le code $e_\sigma$ de la fonction qui l'entrée $e_\sigma$ cherche une extension $\tau_\sigma \succeq \sigma$ telle que $\Phi(\tau_\sigma, e_\sigma)\downarrow$ et assigne $\Phi(\tau_\sigma, e_\sigma)$ à $\Phi_{e_\sigma}(e_\sigma)$. Le processus étant uniforme, on énumère toutes les chaînes de la forme $\tau_\sigma$ dans un ensemble c. e. $W$.

Notons que pour tout $\sigma \prec G$ la fonction de code $e_\sigma$ s'arrête sur l'entrée $e_\sigma$, car il existe au moins une extension de $\sigma$ — à savoir $G$ lui-même — pour laquelle $\Phi$ s'arrête sur l'entrée $e_\sigma$. Comme $n \mapsto \Phi(G, n)$ est DNC, on a dès lors $\Phi(G, e_\sigma) \neq \Phi_{e_\sigma}(e_\sigma)$, et donc $\sigma \prec \tau_\sigma \not\prec G$.

On a ainsi un ensemble c. e. $W$ dont aucun élément n'est un préfixe de $G$, mais qui contient une extension de chaque préfixe de $G$. Il s'ensuit que $G$ n'est pas 1-générique. ∎

La restriction de la généricité permet de construire des ensembles bénéficiant de certains avantages de la généricité tout en n'étant pas trop complexes d'un point de vue calculatoire.

**Exercice 3.22. (⋆)** Montrer par un argument direct qu'aucun ensemble 1-générique n'est calculable. ◇

**Exercice 3.23. (⋆)** Montrer qu'il existe un ensemble 1-générique $\Delta^0_2$. En déduire qu'il existe un ensemble 1-générique de degré low. ◇

Nous allons voir dans la section suivante avec le corollaire 3.34 qu'il existe aussi des ensembles 1-génériques de degrés high. Terminons par un exercice intéressant, qui demande d'élaborer sur les techniques du théorème 3.28, et qui utilise la notion de jointure effective de la définition 4-5.6, mais étendue à une suite dénombrable d'ensembles :

**Définition 3.24.** La *jointure effective* $\bigoplus_{n\in\mathbb{N}} A_n$ d'une suite $(A_n)_{n\in\mathbb{N}}$ d'ensembles est l'ensemble $Y$ tel que $\langle n, m\rangle \in Y$ ssi $m \in A_n$. ♢

Dans les exercices qui suivent, la notation $\bigoplus_{j\neq i} G_j$ correspond donc à la jointure effective de la suite $(G_n)_{n\in\mathbb{N}}$ de laquelle on enlève l'ensemble $G_i$.

**Exercice 3.25.** (⋆⋆) Soit $G = \bigoplus_{n\in\mathbb{N}} G_n$ un ensemble 1-générique. Montrer que $G_i$ est hyperimmune relativement à $\bigoplus_{j\neq i} G_j$ pour tout $i \in \mathbb{N}$. ◇

**Exercice 3.26.** (⋆⋆) (*Miller*). Soit $X = \bigoplus_{n\in\mathbb{N}} X_n$ un ensemble tel que $X_i$ est hyperimmune relativement à $\bigoplus_{j\neq i} X_j$ pour tout $i \in \mathbb{N}$. Montrer que $X$ calcule un ensemble 1-générique. ◇

### 3.2.3. Relativisation des ensembles 1-génériques

Tout comme on a relativisé la 1-généricité faible, on relativise à présent la 1-généricité.

**Définition 3.27 (Jockusch [100]).** Soit $A \in 2^{\mathbb{N}}$. Un ensemble $G \in 2^{\mathbb{N}}$ est 1-*générique* relativement à $A$ si $G \in \mathcal{U} \cup \mathrm{int}(2^{\mathbb{N}} \setminus \mathcal{U})$ pour toute classe $\mathcal{U}$ qui est $\Sigma^0_1(A)$. On dira que $G$ est $n$-générique s'il est 1-générique relativement à $\emptyset^{(n-1)}$, ou de manière équivalente s'il rencontre $W \cup W^{\perp}$ pour tout $W$ ensemble de chaîne $\Sigma^0_n$. ♢

Nous étudierons cette relativisation plus en détail dans la section 5. Voyons pour le moment un premier théorème clef.

**Théorème 3.28**
*Soit $X$ non calculable, et soit $G$ un ensemble* 1*-générique relativement à $X$. Alors, $G \not\geq_T X$.*

Preuve. Soit $\Phi$ une fonctionnelle Turing. Nous construisons $\mathcal{U}$, une classe $\Sigma^0_1(X)$ telle que $Y \in \mathcal{U} \cup \mathrm{int}(\mathbb{N}\setminus\mathcal{U})$ implique $\Phi(Y) \neq X$.

On énumère simplement dans l'ensemble $X$-c. e. qui décrit $\mathcal{U}$, toutes les chaînes $\sigma$ telles que $\exists n\ \Phi(\sigma, n)\!\downarrow \neq X(n)$. Supposons à présent que $\tau$ est une chaîne pour laquelle $[\tau] \subseteq \mathbb{N} \setminus \mathcal{U}$, c'est-à-dire qu'aucune extension $\sigma \succeq \tau$ ne soit telle que $\exists n\ \Phi(\sigma, n)\!\downarrow \neq X(n)$. Montrons alors que, pour tout $Y \succeq \tau$,

on a $\exists n\ \Phi(Y,n)\uparrow$. Supposons que ce ne soit pas le cas. Alors, on peut calculer $X$ de la manière suivante : pour connaître $X(n)$, il suffit de chercher une extension $\sigma \succeq \tau$ telle que $\Phi(\sigma,n)\downarrow$. Par hypothèse, une telle extension existe, et toujours par hypothèse, elle est telle que $\Phi(\sigma,n)\downarrow = X(n)$. Comme c'est vrai pour tout $n$, cela contredit le fait que $X$ est incalculable. Donc, si $[\tau] \subseteq \mathbb{N} \setminus \mathcal{U}$, alors pour tout $Y \succeq \tau$ on a $\exists n\ \Phi(Y,n)\uparrow$. On en déduit que, pour tout $Y \in \mathcal{U} \cup \text{int}(\mathbb{N} \setminus \mathcal{U})$, on a $\Phi(Y) \neq X$. Comme $G$ est 1-générique relativement à $X$, alors $G \in \mathcal{U} \cup \text{int}(\mathbb{N} \setminus \mathcal{U})$, et la même chose est vrai pour toute fonctionnelle $\Phi$. Donc, $G \not\geq_T X$. ∎

Voyons à présent les différentes implications entre les notions de $n$-généricité et $n$-généricité faible.

**Proposition 3.29.** Soit $G$ un ensemble. Alors, pour tout $n > 0$, $G$ faiblement $(n+1)$-générique implique $G$ $n$-générique, lequel implique à son tour $G$ faiblement $n$-générique. Les implications sont strictes. ⋆

Preuve. Si $G$ est faiblement $(n+1)$-générique, il rencontre tout ensemble $\Sigma^0_1(\emptyset^{(n)})$ dense. Pour tout ensemble $\Sigma^0_1(\emptyset^{(n-1)})$ $W$, la réunion $W \cup W^\perp$ est dense et $\Sigma^0_1(\emptyset^{(n)})$, donc rencontre $G$. Ainsi, $G$ est $n$-générique. Si $G$ est $n$-générique, alors pour tout ensemble $\Sigma^0_1(\emptyset^{(n-1)})$, $G$ rencontre $W \cup W^\perp$. Si $W$ est dense, alors $W^\perp = \emptyset$, donc $G$ rencontre $W$. Ainsi, $G$ est faiblement $n$-générique.

Pour voir que la première implication est stricte, il suffit de construire un ensemble $n$-générique $\emptyset^{(n)}$-calculable et de voir qu'aucun ensemble faiblement $(n+1)$-générique n'est $\emptyset^{(n)}$-calculable, car $\{\sigma : \sigma \not\preceq X\}$ est un ensemble de chaînes $\Sigma^0_{n+1}$ dense pour tout ensemble $X$ qui est $\Delta^0_{n+1}$. On pourra consulter l'exercice 3.30 pour un exemple d'ensemble $n$-générique qui n'est pas faiblement $n$-générique. ∎

**Exercice 3.30. (⋆⋆)** Un ensemble $X$ est *approchable par la gauche* relativement à $A$ s'il existe une suite $A$-calculable d'ensembles $(X_s)_{s\in\mathbb{N}}$ telle que $X_s$ est lexicographiquement plus petit que $X_{s+1}$ pour tout $s$, et tel que $\lim_s X_s = X$.

Montrer que pour tout $A$ il existe un ensemble faiblement 1-générique relativement à $A$ et approchable par la gauche relativement à $A$. Montrer qu'aucun ensemble 1-générique relativement à $A$ n'est approchable par la gauche relativement à $A$. ◇

### 3.3. Théorème de Posner/Robinson

Dans un article de 1981, Posner et Robinson étudient les degrés strictement sous l'arrêt, mais dont la jointure permet de calculer l'arrêt. Une version généralisée et moderne de leur théorème principal est la suivante : pour tout ensemble $A$ non calculable — et en particulier aussi « faible » que possible calculatoirement —, il existe un ensemble $G$ tel que $A \oplus G \geqslant_T G'$. Informellement, il existe toujours un ensemble $G$ dont la distance calculatoire entre lui-même et son saut est « réduite » à $A$, et ce pour n'importe quel $A$.

La présentation moderne du théorème est celle de Jockusch et Shore, qui montrent quelque chose de plus général :

**Théorème 3.31 (Jockusch et Shore [102])**
*Soient $A, Z$ des ensembles non calculables. Il existe un ensemble 1-générique $G$ tel que $A \oplus G \geqslant_T Z$. De plus, on peut obtenir $G$ de manière calculable à partir de $A \oplus Z \oplus \emptyset'$.*

Preuve. L'idée est de construire un ensemble 1-générique $G$, qui va encoder $Z$, de manière à ce que $G$ et $A$ permettent de retrouver le déroulement de la construction. La construction elle-même sera calculable en $A \oplus Z \oplus \emptyset'$. On peut supposer sans perte de généralité que $A$ n'est pas un ensemble c. e. (dans le cas contraire, on remplace $A$ par son complémentaire). Soit $(W_e)_{e \in \mathbb{N}}$ une énumération des sous-ensembles $\Sigma^0_1$ de $2^{<\mathbb{N}}$.

On définit $\sigma_0 = \epsilon$, le mot vide. Supposons $\sigma_e$ défini. On considère l'ensemble

$$D_e = \{m : \exists \tau \text{ tel que } \sigma_e Z(e) 0^m 1 \tau \in W_e\}.$$

Notons que $D_e$ est un ensemble c. e. En particulier, comme $A$ n'est pas c. e., il existe $m \in D_e$, avec $m \notin A$, ou alors il existe $m \notin D_e$ avec $m \in A$. On considère le plus petit $m$ tel que l'on est dans un cas ou dans l'autre. Notons que $\emptyset' \oplus A$ permet de trouver uniformément cet entier $m$.

Dans le premier cas, on définit $\sigma_{e+1}$ comme étant $\sigma_e Z(e) 0^m 1 \tau$ pour la première chaîne $\tau$ telle que $\sigma_e Z(e) 0^m 1 \tau$ est énumérée dans $W_e$. Dans le deuxième cas, on définit $\sigma_{e+1}$ comme étant $\sigma_e Z(e) 0^m 1$. Notons que dans ce cas aucune chaîne de $W_e$ ne peut étendre $\sigma_{e+1}$. On définit $G$ comme étant $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ Cela termine la construction.

Il est clair que $G$ est 1-générique et calculable en $A \oplus Z \oplus \emptyset'$. Comment fait-on à présent pour calculer $Z$ à partir de $G \oplus A$ ? Supposons que l'on connaisse la chaîne $\sigma_e$. On connaît alors nécessairement le $e$-ième bit de $Z$ : il s'agit du bit $i$ tel que $\sigma_e i \prec G$. On peut ensuite trouver $\sigma_{e+1}$ de la manière suivante : on regarde le nombre $m$ de 0 qui suit $\sigma_e i$ dans $G$.

Si $m \in A$, cela signifie que $\sigma_{e+1} = \sigma_e i 0^m 1$. Si $m \notin A$, cela signifie que $\sigma_{e+1} = \sigma_e i 0^m 1 \tau$ pour la première chaîne $\tau$ que l'on trouve dans $W_e$. Trouver $\tau$ est alors un procédé calculable. On peut donc dans tous les cas trouver $\sigma_{e+1}$, et en répétant le processus, calculer $Z$ à partir de $A \oplus G$. ∎

**Corollaire 3.32 (Posner et Robinson [180])**
*Soit $A$ un ensemble non calculable. Il existe un ensemble $G$ tel que*

$$A \oplus G \geqslant_T G'.$$

*Si $A$ est $\Delta^0_2$, on a alors $A \oplus G \equiv_T G'$.*

Preuve. On applique le théorème précédent avec $Z = \emptyset'$. On a donc un ensemble 1-générique $G$ tel que $G \leqslant_T \emptyset' \oplus A$ et tel que $G \oplus A \geqslant_T \emptyset'$. En utilisant le fait que les 1-génériques sont low généralisés, on a donc

$$G \oplus A \geqslant_T G \oplus \emptyset' \equiv_T G'.$$

Il est clair que si $A$ est $\Delta^0_2$, alors $A \oplus G \equiv_T G'$. ∎

Un autre corollaire intéressant est le théorème d'inversion du saut : tout ensemble qui calcule l'arrêt peut être vu comme le degré Turing du saut d'un ensemble.

**Corollaire 3.33 (Théorème d'inversion du saut, Friedberg [62])**
*Soit $Z \geqslant_T \emptyset'$. Il existe un ensemble $G$ tel que $Z \equiv_T G'$.*

Preuve. On applique le théorème précédent avec $Z$ et $A = \emptyset'$. On a donc un ensemble 1-générique $G$ tel que

$$G \oplus \emptyset' \leqslant_T Z \oplus \emptyset' \equiv_T Z \quad \text{et tel que} \quad G \oplus \emptyset' \geqslant_T Z.$$

Usant du fait que les 1-génériques sont low généralisés, on a donc $G' \equiv_T Z$. ∎

Le théorème 3.31 permet également de déduire l'existence d'ensembles high et non Turing complets, et même non DNC.

**Corollaire 3.34**
*Il existe un ensemble high, non DNC et en particulier non Turing complet.*

Preuve. Il suffit d'appliquer le théorème 3.31 pour trouver un ensemble 1-générique $G$ tel que $\emptyset' \oplus G \geqslant_T \emptyset''$, ce qui implique $G' \geqslant_T \emptyset''$. Notons que comme $G$ est 1-générique, il n'est pas de degré DNC, et ne calcule en particulier pas $\emptyset'$. ∎

### 3.4. Maigreur/co-maigreur des propriétés calculatoires

Revenons un moment aux notions topologiques de classes maigres et de classes co-maigres introduites par Baire. Nous avons déjà mentionné que toute classe n'était pas forcément maigre ou co-maigre. C'est en revanche le cas pour les classes ayant de bonnes propriétés de clôture, et en particulier pour toutes celles dites boréliennes (nous définirons précisément ce terme dans le chapitre 17) et closes par équivalence Turing. Ce sera en particulier le cas de toutes les notions de calculabilité que nous verrons, et nous nous poserons la question de savoir si ces dernières sont maigres ou co -maigres. Aussi une classe est-elle sans perte de généralité co-maigre si elle contient tout élément *suffisamment générique*. En pratique, la 1-généricité est suffisante pour les différentes propriétés calculatoires vues jusqu'ici.

Nous avons les degrés high —les low en tant que classe dénombrable ne nous intéressent pas— les degrés calculatoirement dominés vs hyperimmunes, et enfin les degrés DNC et PA. La classe des ensembles de chacun de ces degrés est-elle maigre ou co-maigre ? Nous avons déjà la réponse en ce qui concerne les degrés calculatoirement dominés et hyperimmunes.

**Proposition 3.35.** La classe des ensembles calculatoirement dominés est maigre. Celle des ensembles hyperimmunes est co-maigre. ⋆

Preuve. D'après le théorème 3.2, si $G$ est faiblement 1-générique, alors il n'est pas de degré calculatoirement dominé. ∎

**Proposition 3.36.** La classe des ensembles DNC est maigre, ainsi bien sûr que la classe des ensembles PA. ⋆

Preuve. D'après le théorème 3.21, si $X$ est 1-générique,alors il n'est pas de degré DNC, hors la classe des ensembles 1-génériques est co-maigre. ∎

Nous nous attaquons à présent aux degrés high. Nous utilisons pour cela notre théorème 3.28 d'évitement de cône.

**Proposition 3.37.** Si $X$ est non calculable, alors la classe des ensembles qui calculent $X$ est maigre. ⋆

Preuve. La classe des ensembles 1-génériques relativement à $X$ est une intersection d'ouverts denses, et est donc co-maigre. Par le théorème 3.28, aucun d'entre eux ne calcule $X$, et la classe des ensemble calculant $X$ est donc dans son complémentaire, une classe maigre. ∎

Nous pouvons finalement montrer, en combinant ce que nous avons vu, que la classe des ensembles high est elle aussi maigre.

**Proposition 3.38.** La classe des ensembles high est maigre. ⋆

Preuve. On utilise pour cela une version relativisée du théorème 3.28 : si $X$ n'est pas $\emptyset'$-calculable, alors pour tout ensemble $G$ qui est 1-générique relativement à $X \oplus \emptyset'$ on a $G \oplus \emptyset' \not\geqslant_T X$. Cette version relativisée se montre de la même manière et ne présente pas de difficulté particulière.
On utilise à présent le théorème 3.20 : si l'ensemble $G$ est 1-générique, on a alors $G' \leqslant_T G \oplus \emptyset'$. On en déduit que si $G$ est 1-générique, alors $G$ est high ssi $G \oplus \emptyset' \geqslant_T \emptyset''$. Par ailleurs, d'après la version relativisée du théorème 3.28, aucun 1-générique relativement à $\emptyset''$ n'est tel que $G \oplus \emptyset' \geqslant_T \emptyset''$. Comme tout ensemble 1-générique relativement à $\emptyset''$ est aussi 1-générique, on a donc qu'aucun 1-générique relativement à $\emptyset''$ n'est high. La classe des ensembles high est donc maigre. ∎

# 4. Forcing $\Sigma^0_n/\Pi^0_n$

Le concept de 1-généricité peut être vu comme un cadre formel effectif autour de la méthode des extensions finies, qui permet pour résumer de contrôler l'arrêt ou non de fonctionnelles, c'est-à-dire de contrôler la valeur de vérité de prédicats $\Sigma^0_1$ ou $\Pi^0_1$. Il s'agit d'un premier niveau de forcing : étant donné $\mathcal{R}_e$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$, d'après le corollaire 3.18 toute chaîne $\sigma$ admet une extension $\tau \succeq \sigma$ telle que tout ensemble $X \in [\tau]$ satisfait $\mathcal{R}_e$ ou telle que tout ensemble $X \in [\tau]$ satisfait $\neg\mathcal{R}_e$.

À partir du niveau $\Sigma^0_2/\Pi^0_2$, les choses deviennent plus complexes et la méthode des extensions finies ne peut plus fonctionner de la même manière, comme en témoigne l'exemple suivant.

**Exemple 4.1.** Considérons le contrat $\mathcal{R}$ : « $\exists x \forall y \geqslant x\ G(y) = 0$ », qui exprime la finitude de l'ensemble $G$. Pour tout $\sigma \in 2^{<\mathbb{N}}$, $[\sigma]$ contient à la fois un ensemble fini et un ensemble infini. Il n'existe donc aucun cylindre dont tous les éléments satisfont $\mathcal{R}$ ou dont tous les éléments satisfont $\neg\mathcal{R}$.

**Point sur les contrats**

Nous nous attaquons à présent aux contrats $\Sigma^0_n$ (resp. $\Pi^0_n$) c'est-à-dire aux contrats portant sur un ensemble $G$ et qui s'expriment par une formule $\Sigma^0_n$ (resp. $\Pi^0_n$) de l'arithmétique du second ordre, avec $G$ comme variable d'ensemble libre. De manière équivalente, un contrat est $\Sigma^0_n$ s'il existe une fonctionnelle $\Phi_e(G, x_1, \dots, x_{n-1})$ telle que les ensembles $G$ satisfaisant le contrat sont ceux tels que :

$$\exists x_1\ \forall x_2 \dots \forall x_{n-1}\ \Phi_e(G, x_1, x_2, \dots, x_{n-1})\downarrow \quad \text{pour } n \text{ impair}$$
$$\exists x_1\ \forall x_2 \dots \exists x_{n-1}\ \Phi_e(G, x_1, x_2, \dots, x_{n-1})\uparrow \quad \text{pour } n \text{ pair.}$$

Les contrats $\Pi^0_n$ ont l'équivalence analogue en commençant par une quantification universelle. Par convention, la négation $\neg\mathcal{R}$ devant une formule $\Sigma^0_n$ (resp. $\Pi^0_n$) ne sera pas considérée comme un symbole du langage, mais comme une opération de transformation $\mathcal{R}$, qui inverse les quantificateurs, et remplace le prédicat $\Delta^0_0$ final par sa négation, pour faire de $\neg\mathcal{R}$ une formule $\Pi^0_n$ (resp. $\Sigma^0_n$).

Nous devons donc abstraire un peu les choses afin de donner une définition plus générale du forcing permettant de contrôler la valeur de vérité de prédicats de complexité arbitraire. Introduisons avant de commencer une définition qui sera utilisée dans les preuves à venir.

**Définition 4.2.** Un ensemble $D \subseteq 2^{<\mathbb{N}}$ est *dense sous la chaîne* $\sigma$ si pour tout $\tau \succeq \sigma$, il existe un $\rho \succeq \tau$ tel que $\rho \in D$. ♢

### 4.1. L'approche sémantique

La relation de forcing sera définie par induction sur $n$, entre les chaînes finies et les prédicats $\Sigma^0_n$. Un des objectifs sera alors de conserver la propriété suivante.

(D) : L'ensemble des chaînes forçant $\mathcal{R}$ ou forçant $\neg\mathcal{R}$ est dense.

Afin d'examiner ce dont nous avons besoin, reprenons l'exemple précédent, à savoir un contrat $\mathcal{R}$ $\Sigma^0_2$ arbitraire $\ll \exists x\, \Phi(G,x)\!\uparrow \gg$, et regardons ce qui ne fonctionne pas quand on essaye de prouver la densité de l'ensemble $Q \subseteq 2^{<\mathbb{N}}$ des chaînes dont tous les éléments satisfont $\mathcal{R}$ ou dont tous les éléments satisfont $\neg\mathcal{R}$.

Soit $\sigma \in 2^{<\mathbb{N}}$ une chaîne. Faisons une analyse de cas.

▷ Cas 1. Il existe une extension $\tau \succeq \sigma$ et un entier $x \geqslant 1$ tel que pour tout $\rho \succeq \tau$, $\Phi(\rho,x)\uparrow$. Dans ce cas, par la propriété de l'usage, pour tout $A \in [\tau]$, $\Phi(A,x)\uparrow$. Il s'ensuit que $\mathcal{R}$ est satisfait pour tout $A \in [\tau]$, donc que $\tau \in Q$.

▷ Cas 2. Pour toute extension $\tau \succeq \sigma$ et tout $x \in \mathbb{N}$, il existe une chaîne $\rho \succeq \tau$ telle que $\Phi(\rho,x)\!\downarrow$. C'est dans ce cas que notre tentative échoue. Pourtant, intuitivement, dans ce cas, nous devrions être capables de poursuivre la construction d'une suite tout en s'assurant que l'ensemble résultant satisfasse $\neg\mathcal{R}$. En effet, quel que soit l'état d'avancement de la construction, nous nous retrouvons avec une chaîne $\tau \succeq \sigma$, donc par hypothèse, pour tout $x \in \mathbb{N}$, il est toujours possible de trouver une extension $\rho \succeq \tau$ telle que $\Phi(G,x)\!\downarrow$ pour tout $G \in [\rho]$ : pour tout $x$, l'ensemble $Q_x$ des chaînes $\rho$ telles que $\Phi(\rho,x)\downarrow$ est dense sous $\sigma$, c'est-à-dire que pour tout $\tau \succeq \sigma$ il

existe $\rho \succeq \tau$ telle que $\rho \in Q_x$. Par conséquent, si un ensemble suffisamment générique $G$ étend $\sigma$, alors il rencontrera chacun des $Q_x$. On aura donc $\forall x\ \Phi(G, x)\downarrow$, autrement dit $G$ satisfera $\neg\mathcal{R}$.

Cette analyse motive donc la définition suivante pour le forcing de Cohen.

**Définition 4.3.** Une chaîne $\sigma$ *force sémantiquement* un contrat $\mathcal{R}$, auquel cas on notera $\sigma \Vdash \mathcal{R}$, si tout ensemble « suffisamment générique » qui étend $\sigma$ satisfait $\mathcal{R}$ : il existe une suite dénombrable d'ensembles de chaînes denses $(W_n)_{n\in\mathbb{N}}$ telle que si $G \in [\sigma]$ rencontre chaque $W_n$, alors le contrat sera satisfait pour $G$. ♢

Notons que la relation $\Vdash$ est plus générale que la relation $\Vdash^*$ pour les contrats $\Sigma^0_1$ et $\Pi^0_1$. Par exemple, la chaîne vide $\epsilon$ force sémantiquement le contrat $\mathcal{R} : \ll \exists x\ G(x) = 1 \gg$, car l'ensemble des chaînes contenant un 1 est dense, et donc tout ensemble suffisamment générique satisfera $\mathcal{R}$. En revanche, la suite infinie de zéros $0^\infty$ appartient à $[\epsilon]$ et ne satisfait pas le contrat $\mathcal{R}$.

### 4.2. L'approche syntaxique

La définition 4.3 est simple à exprimer dans le langage naturel, mais sort de la hiérarchie arithmétique : une traduction directe demande de quantifier existentiellement sur toutes les suites dénombrables d'ensembles de chaînes denses, puis de quantifier universellement sur les ensembles génériques pour ces ensembles de chaînes. Nous allons définir une relation beaucoup plus simple syntaxiquement parlant, qui permettra de raisonner plus facilement et notamment de prouver les propriétés essentielles que l'on attend de la relation de forcing, à savoir qu'elle soit close par extension et que l'ensemble des chaînes forçant un contrat ou sa négation soit dense.

Définissons à présent une relation syntaxique de forcing pour tout contrat arithmétique, par induction sur ses quantifications. Le premier niveau sera celui auquel nous sommes déjà habitué : soit $\mathcal{R}$ un contrat $\Sigma^0_1$ — et donc de la forme $\Phi_e(G)\downarrow$ —, alors une chaîne $\sigma$ force $\mathcal{R}$ si $\Phi_e(\sigma)\downarrow$, et donc si tout $X \in [\sigma]$ satisfait le contrat. Il en va de même pour les contrats $\Pi^0_1$.

**Définition 4.4**

(1) $\sigma \Vdash^* \mathcal{R}$ pour $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$ ssi tout $X \in [\sigma]$ satisfait $\mathcal{R}$.

(2) $\sigma \Vdash^* \exists x\mathcal{R}(x)$ pour $\mathcal{R}(x)$ un contrat $\Pi^0_k$ pour $k > 0$ avec variable libre $x$ ssi il existe un $n \in \mathbb{N}$ tel que $\sigma \Vdash^* \mathcal{R}(n)$.

(3) $\sigma \Vdash^* \forall x\mathcal{R}(x)$ pour $\mathcal{R}(x)$ un contrat $\Sigma^0_k$ pour $k > 0$ avec variable libre $x$ ssi pour tout $\tau \succeq \sigma$ et tout $n \in \mathbb{N}$, $\tau \nVdash^* \neg\mathcal{R}(n)$. ♢

Avant toute chose, remarquons que (3) de la définition précédente admet une formulation équivalente, parfois plus adaptée à ce que l'on veut démontrer.

**Lemme 4.5.** Soit $\mathcal{R}$ un contrat $\Pi^0_n$, on a $\sigma \Vdash^* \mathcal{R}$ ssi $\forall \tau \succeq \sigma\ \tau \nVdash^* \neg\mathcal{R}$. ⋆

Preuve. Il s'agit d'une simple reformulation de la définition.

Cas 1. Le contrat $\mathcal{R}$ est $\Pi^0_1$, de la forme $\Phi(G)\uparrow$. Alors, $\sigma \Vdash^* \mathcal{R}$ ssi $\Phi(X)\uparrow$ pour tout $X \in [\sigma]$ ssi $\Phi(X)\uparrow$ pour tout $\tau \succeq \sigma$ et tout $X \in [\tau]$ ssi pour tout $\tau \succeq \sigma$, il existe $X \in [\tau]$ tel que $\Phi(X)\uparrow$ (par la propriété de l'usage) ssi pour tout $\tau \succeq \sigma$, $\tau \nVdash^* \Phi(G)\downarrow$.

Cas 2. Le contrat $\mathcal{R}$ est $\Pi^0_{k+1}$, de la forme $\forall x \mathcal{Q}(x)$ pour $\mathcal{Q}(x)$ un contrat $\Sigma^0_k$ pour $k \geqslant 1$. Alors, $\sigma \Vdash^* \mathcal{R}$ ssi pour tout $\tau \succeq \sigma$ et tout $n \in \mathbb{N}$, $\tau \nVdash^* \neg\mathcal{Q}(n)$ ssi $\forall \tau \succeq \sigma\ \tau \nVdash^* \exists x\ \neg\mathcal{Q}(x)$ ssi $\forall \tau \succeq \sigma\ \tau \nVdash^* \neg\mathcal{R}$. ∎

La première propriété que l'on attend d'une relation de forcing est sa clôture par extension. En effet, « $\sigma$ force $\mathcal{R}$ » signifie que la propriété $\mathcal{R}$ est déjà décidée sur l'objet final construit, ce qui ne doit pas changer au cours des étapes suivantes de la construction.

**Proposition 4.6.** Soient $\sigma, \tau \in 2^{<\mathbb{N}}$ et soit $\mathcal{R}$ un contrat arithmétique. Si $\sigma \Vdash^* \mathcal{R}$ et $\sigma \preceq \tau$, alors $\tau \Vdash^* \mathcal{R}$. ⋆

Preuve. Par induction sur la complexité arithmétique du contrat.

Cas 1. Le contrat $\mathcal{R}$ est $\Sigma^0_1$ ou $\Pi^0_1$. Par définition, pour tout $X \in [\sigma]$, $\mathcal{R}(X)$ est vrai. Comme $\tau \succeq \sigma$, alors $[\tau] \subseteq [\sigma]$, donc pour tout $X \in [\tau]$, $\mathcal{R}(X)$ est vrai. Ainsi, $\tau \Vdash^* \mathcal{R}$.

Cas 2. Le contrat $\mathcal{R}$ est de la forme $\exists x \mathcal{S}(x)$ pour $S(x)$ un contrat $\Pi^0_k$ pour $k > 0$. Par définition, il existe un $n \in \mathbb{N}$ tel que $\sigma \Vdash^* \mathcal{S}(n)$. Par hypothèse d'induction, $\tau \Vdash^* \mathcal{S}(n)$, donc $\tau \Vdash^* \exists x \mathcal{S}(x)$.

Cas 3. Le contrat $\mathcal{R}$ est de la forme $\forall x \mathcal{S}(x)$ pour $S(x)$ un contrat $\Sigma^0_k$ pour $k > 0$. D'après le lemme 4.5, $\sigma \Vdash^* \mathcal{R}$ implique $\forall \rho \succeq \sigma\ \rho \nVdash^* \neg\mathcal{R}$. Si $\tau \succeq \sigma$, alors on a aussi $\forall \rho \succeq \tau\ \rho \nVdash^* \neg\mathcal{R}$, ce qui encore d'après le lemme 4.5 implique $\tau \Vdash^* \mathcal{R}$. ∎

La seconde propriété, et peut-être la plus importante, est la densité de l'ensemble des chaînes forçant un contrat ou sa négation. La proposition 4.7 signifie en particulier que la valeur de vérité de tout contrat arithmétique sera décidée au bout d'un moment fini de la construction. C'est ce qui donne toute la puissance de la relation de forcing.

**Proposition 4.7.** Soit $\mathcal{R}$ un contrat arithmétique. L'ensemble

$$\{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash^* \mathcal{R} \text{ ou } \sigma \Vdash^* \neg\mathcal{R}\}$$

est dense. ⋆

Preuve. On peut supposer sans perte de généralité que $\mathcal{R}$ est $\Sigma^0_n$ (dans le cas inverse, on répète l'argument avec $\neg\mathcal{R}$). Soit $\sigma$ une chaîne. D'après le lemme 4.5, soit $\tau \Vdash^* \mathcal{R}$ pour une extension $\tau \succeq \sigma$, soit $\sigma \Vdash^* \neg\mathcal{R}$. ∎

Une dernière propriété que l'on attend également est bien entendu la validité de la définition de la relation de forcing, c'est-à-dire que si une chaîne $\sigma$ force un contrat, alors ce contrat sera effectivement satisfait pour n'importe quel ensemble suffisamment générique qui étend $\sigma$. Insistons encore sur ce que signifie être « suffisamment générique » dans ce contexte : il existe une suite dénombrable d'ensembles de chaînes denses $(W_n)_{n\in\mathbb{N}}$ telle que si $G \in [\sigma]$ rencontre chaque $W_n$, alors le contrat sera satisfait pour $G$.

**Proposition 4.8.** Soit $\mathcal{R}$ un contrat arithmétique et soit $\sigma \in 2^{<\mathbb{N}}$.
Si $\sigma \Vdash^* \mathcal{R}$, alors $\sigma \Vdash \mathcal{R}$ : si $G \in [\sigma]$ est suffisamment générique, alors $\mathcal{R}$ est satisfait pour $G$. ⋆

Preuve. Par induction sur la complexité arithmétique du contrat.

Cas 1. Le contrat $\mathcal{R}$ est $\Sigma^0_1$ ou $\Pi^0_1$. Supposons que $\sigma \Vdash^* \mathcal{R}$. Par définition, tout ensemble $G \in [\sigma]$ satisfait $\mathcal{R}$, donc *a fortiori* tout ensemble suffisamment générique $G \in [\sigma]$. Ainsi, $\sigma \Vdash \mathcal{R}$.

Cas 2. Le contrat $\mathcal{R}$ est de la forme $\exists x \mathcal{S}(x)$ pour $S(x)$ un contrat $\Pi^0_k$ pour $k > 0$. Supposons $\sigma \Vdash^* \exists x \mathcal{S}(x)$. Par définition, il existe un $n \in \mathbb{N}$ tel que $\sigma \Vdash^* \mathcal{S}(n)$. Par hypothèse d'induction, $\sigma$ force $\mathcal{S}(n)$, donc $\sigma \Vdash \exists x \mathcal{S}(x)$.

Cas 3. Le contrat $\mathcal{R}$ est de la forme $\forall x \mathcal{S}(x)$ pour $S(x)$ un contrat $\Sigma^0_k$ pour $k > 0$. Supposons que $\sigma \Vdash^* \forall x \mathcal{S}(x)$. Par définition, pour tous $\tau \succeq \sigma$ et $n \in \mathbb{N}$, $\tau \nVdash^* \neg\mathcal{S}(x)$. Par la proposition 4.7, pour tout $n$, l'ensemble

$$D_n = \{\tau \in 2^{<\mathbb{N}} : \tau \Vdash^* \mathcal{S}(n)\}$$

est dense sous $\sigma$ : pour tout $\tau \succeq \sigma$ il existe $\rho \succeq \tau$ tel que $\rho \Vdash^* \mathcal{S}(n)$. Soit $G$ un ensemble suffisamment générique étendant $\sigma$. On suppose en particulier que le niveau de généricité de $G$ garanti qu'il rencontre chaque ensemble $D_n$. Donc, pour tout $n$, il existe un préfixe $\tau_n \prec G$ tel que $\tau_n \Vdash^* \mathcal{S}(n)$. Pour une telle chaîne $\tau_n$, par hypothèse d'induction, il existe une suite dénombrable d'ensemble de chaînes dense $(D_{n,m})_{m\in\mathbb{N}}$ telle que si $G \in [\tau_n]$ rencontre chaque $D_{n,m}$, alors il satisfait $\mathcal{S}(n)$. On a bien $G \in [\tau_n]$, et comme $G$ est suffisamment générique, il rencontre chaque $D_{n,m}$, et satisfait donc $\mathcal{S}(n)$. Comme c'est le cas pour tout $n$, alors $G$ satisfait $\forall x \mathcal{S}(x)$, donc $\sigma \Vdash \forall x \mathcal{S}(x)$. ∎

Le cœur du forcing se trouve sans doute dans la précédente proposition, et en particulier dans le cas 3 de sa preuve : c'est là que s'exerce le mécanisme de la relation $\sigma \Vdash^* \mathcal{R}$, qui garantit que si $G$ appartient à $[\sigma]$ et si $G$ est suffisamment générique, alors le contrat $\mathcal{R}$ sera satisfait pour $G$. Il s'agit en réalité d'une modification sophistiquée de la méthode des extensions finies, le point clef étant le suivant : peu importe le préfixe $\sigma$ de $G$ que l'on a construit jusqu'ici, on peut étendre $\sigma$ pour rencontrer n'importe quel ensemble de chaînes dense fixé à l'avance.

Nous voyons à présent dans la section suivante que cette idée n'est pas nouvelle : le mathématicien René Baire avait déjà formalisé l'ensemble de ces mécanismes au début du XX$^{\text{e}}$ siècle, notamment via le théorème suivant : « toute classe borélienne a la propriété de Baire. »

### 4.3. L'approche topologique : la propriété de Baire

Une technique importante pour l'étude d'objets complexes en mathématiques consiste à se ramener à des objets plus simples tout en contrôlant la marge d'erreur d'approximation. En particulier, dans l'étude des ensembles, que cela soit du point de vue de la théorie de la mesure ou de la théorie des catégories, il existe un certain nombre de théorèmes de la forme « tout ensemble complexe est équivalent à un ensemble simple modulo une quantité négligeable d'éléments ». En théorie de la mesure par exemple, nous avons les trois principes de Littlewood [145], qui énoncent que tout ensemble mesurable est « presque » une réunion finie d'intervalles, toute fonction est « presque » continue, et toute suite convergente est « presque » uniformément convergente. Dans ce contexte « presque » signifie : « sauf sur un ensemble de mesure inférieure à $\varepsilon$ pour $\varepsilon$ aussi petit que l'on souhaite. »

La propriété de Baire exprime le fait qu'une classe $S$ est « presque » ouverte, où presque signifie dans ce contexte : « sauf sur une classe maigre. »

> **Définition 4.9.** Une classe $\mathcal{B} \subseteq 2^{\mathbb{N}}$ a *la propriété de Baire* s'il existe un ouvert $\mathcal{U} \subseteq 2^{\mathbb{N}}$ tel que $\mathcal{B} \Delta \mathcal{U}$ est une classe maigre. Ici, $\mathcal{B} \Delta \mathcal{U}$ est la classe des éléments sur lesquels $\mathcal{B}$ et $\mathcal{U}$ ne coïncident pas, c'est-à-dire que
> $$(\mathcal{B} \setminus \mathcal{U}) \cup (\mathcal{U} \setminus \mathcal{B}).$$ ♢

Considérons un contrat $\mathcal{R}$, et soient $\mathcal{B}_{\mathcal{R}}$ et $\mathcal{B}_{\neg\mathcal{R}}$ les classes des éléments qui respectivement satisfont et ne satisfont pas ce contrat (en particulier, $\mathcal{B}_{\mathcal{R}} \cap \mathcal{B}_{\neg\mathcal{R}} = \emptyset$ et $\mathcal{B}_{\mathcal{R}} \cup \mathcal{B}_{\neg\mathcal{R}} = 2^{\mathbb{N}}$). Supposons que $\mathcal{B}_{\mathcal{R}}$ et $\mathcal{B}_{\neg\mathcal{R}}$ aient tous les deux la propriété de Baire, et fixons deux ouverts $\mathcal{U}_{\mathcal{R}}$ et $\mathcal{U}_{\neg\mathcal{R}}$ tels que $\mathcal{B}_{\mathcal{R}} \Delta \mathcal{U}_{\mathcal{R}}$ et $\mathcal{B}_{\neg\mathcal{R}} \Delta \mathcal{U}_{\neg\mathcal{R}}$ soient tous les deux maigres. Cela signifie qu'il existe une réunion dénombrable de fermés d'intérieur vide contenant $\mathcal{B}_{\mathcal{R}} \Delta \mathcal{U}_{\mathcal{R}}$ et $\mathcal{B}_{\neg\mathcal{R}} \Delta \mathcal{U}_{\neg\mathcal{R}}$. Par passage au complémentaire, il existe une

intersection dénombrable d'ouverts denses $\bigcap_n \mathcal{U}_n$ telle que pour tout $X$ qui lui appartient, on a d'un côté $X \in \mathcal{B}_\mathcal{R}$ ssi $X \in \mathcal{U}_\mathcal{R}$, et de l'autre $X \in \mathcal{B}_{\neg\mathcal{R}}$ ssi $X \in \mathcal{U}_{\neg\mathcal{R}}$.

En particulier, $X \in \mathcal{B}_\mathcal{R}$ ssi il existe un préfixe $\sigma \prec X$ tel que $[\sigma] \subseteq \mathcal{U}_\mathcal{R}$. On dira alors que $\sigma$ force le contrat $\mathcal{R}$. Notons que si $X \notin \mathcal{B}_\mathcal{R}$, alors $X \in \mathcal{B}_{\neg\mathcal{R}}$ et il existe à ce moment un préfixe $\sigma \prec X$ tel que $[\sigma] \subseteq \mathcal{U}_{\neg\mathcal{R}}$. À ce moment, $\sigma$ force le contrat $\neg\mathcal{R}$. La définition suivante est simplement une reformulation de la définition 4.3 qui définit le forcing sémantique.

**Définition 4.10.** Soit $\mathcal{R}$ un contrat. On dit que $\sigma$ *force sémantiquement* le contrat $\mathcal{R}$, et l'on écrit $\sigma \Vdash \mathcal{R}$ si $[\sigma] \setminus \mathcal{B}_\mathcal{R}$ est une classe maigre, où $\mathcal{B}_\mathcal{R}$ est la classe des éléments qui satisfont $\mathcal{R}$. ♢

Comme mentionné précédemment, la propriété fondamentale que l'on attend de la relation de forcing pour un contrat est la suivante.

(D) : L'ensemble des chaînes forçant $\mathcal{R}$ ou forçant $\neg\mathcal{R}$ est dense.

Nous commençons donc par une caractérisation des contrats pour lesquels cette propriété est vraie, à l'aide de la propriété de Baire.

**Proposition 4.11.** Soit $\mathcal{R}$ un contrat, et soit $\mathcal{B}_\mathcal{R}$ la classe des éléments qui satisfont $\mathcal{R}$. Les énoncés suivants sont équivalents :

(1) $\mathcal{B}_\mathcal{R}$ a la propriété de Baire ;

(2) $\{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \mathcal{R} \text{ ou } \sigma \Vdash \neg\mathcal{R}\}$ est dense. ⋆

Preuve. Soient $U_\mathcal{R} = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \mathcal{R}\}$ et $U_{\neg\mathcal{R}} = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \neg\mathcal{R}\}$.

(1) ⇒ (2). Supposons que $\mathcal{B}_\mathcal{R}$ ait la propriété de Baire. Soit $\mathcal{U}$ un ouvert tel que $\mathcal{B}_\mathcal{R} \Delta \mathcal{U}$ est maigre. En particulier, $\mathcal{U} \setminus \mathcal{B}_\mathcal{R}$ est maigre, donc $\mathcal{U} \subseteq [U_\mathcal{R}]$. De plus, $\mathcal{B}_\mathcal{R} \setminus \mathcal{U} = (2^\mathbb{N} \setminus \mathcal{U}) \setminus \mathcal{B}_{\neg\mathcal{R}}$ est maigre, donc $\mathrm{int}(2^\mathbb{N} \setminus \mathcal{U}) \setminus \mathcal{B}_{\neg\mathcal{R}}$ est maigre. Il s'ensuit que $\mathrm{int}(2^\mathbb{N} \setminus \mathcal{U}) \subseteq [U_{\neg\mathcal{R}}]$. Comme $\mathcal{U} \cup \mathrm{int}(2^\mathbb{N} \setminus \mathcal{U})$ est dense dans $2^\mathbb{N}$, il en est de même de $[U_\mathcal{R}] \cup [U_{\neg\mathcal{R}}]$, donc par l'exercice 2.8, $U_\mathcal{R} \cup U_{\neg\mathcal{R}}$ est dense dans $2^{<\mathbb{N}}$.

(2) ⇒ (1). Supposons que $U_\mathcal{R} \cup U_{\neg\mathcal{R}}$ soit dense dans $2^{<\mathbb{N}}$. Par densité, le complémentaire de $[U_\mathcal{R}] \cup [U_{\neg\mathcal{R}}]$ est un fermé d'intérieur vide, donc maigre. Montrons que $\mathcal{B}_R \Delta [U_\mathcal{R}]$ est maigre. Par définition, pour tout $\sigma \in U_\mathcal{R}$, la classe $[\sigma] \setminus \mathcal{B}_R$ est maigre, donc $[U_\mathcal{R}] \setminus \mathcal{B}_R$ est réunion dénombrable de classes maigres, donc est maigre. Par le même raisonnement, si $\mathcal{B}_{\neg\mathcal{R}}$ est la classe des éléments qui satisfont $\neg\mathcal{R}$, alors $[U_{\neg\mathcal{R}}] \setminus \mathcal{B}_{\neg\mathcal{R}}$ est maigre. Comme $[U_{\neg\mathcal{R}}] \setminus \mathcal{B}_{\neg\mathcal{R}}$ et $\mathcal{B}_R \cap [U_{\neg\mathcal{R}}]$ sont égaux, et comme le complémentaire de $[U_\mathcal{R}] \cup [U_{\neg\mathcal{R}}]$ est maigre, alors $\mathcal{B}_R \setminus [U_\mathcal{R}]$ est maigre. Ainsi, $\mathcal{B}_R \Delta [U_\mathcal{R}]$ est maigre. ∎

La relation de forcing syntaxique impliquant la relation sémantique, nous pouvons en déduire la densité de la relation de forcing sémantique pour les contrats arithmétiques.

**Lemme 4.12.** Soit $\mathcal{R}$ est un contrat arithmétique. L'ensemble

$$\{\sigma \in 2^{<\mathbb{N}} \ : \ \sigma \Vdash \mathcal{R} \vee \sigma \Vdash \neg\mathcal{R}\}$$

est dense. ⋆

Preuve. Pour tout $\sigma$, il existe une extension $\tau \succeq \sigma$ telle que $\tau \Vdash^* \mathcal{R}$ ou telle que $\tau \Vdash^* \neg\mathcal{R}$. Si $\tau \Vdash^* \mathcal{R}$, alors $\tau \Vdash \mathcal{R}$, et si $\tau \Vdash^* \neg\mathcal{R}$, alors $\tau \Vdash \neg\mathcal{R}$. ∎

Le corollaire suivant découle directement de la caractérisation des classes ayant la propriété de Baire.

**Corollaire 4.13**
*Soit $\mathcal{R}$ est un contrat arithmétique. La classe $\mathcal{B}_\mathcal{R}$ des éléments qui satisfont $\mathcal{R}$, a la propriété de Baire.*

Preuve. Immédiat par le lemme 4.12 et la proposition 4.11. ∎

Nous développerons dans le chapitre 17 et la section 27-3.1 la théorie des classes boréliennes, qui formalise et généralise la notion de contrats arithmétiques, et qui ont toutes la propriété de Baire.

Terminons cette section en clarifiant les liens entre la relation de forcing sémantique et la relation syntaxique $\Vdash^*$, que nous avons définies plus haut.

**Théorème 4.14**
*Soit $\mathcal{R}$ un contrat arithmétique. Alors, $\sigma \Vdash \mathcal{R}$ si, et seulement si, l'ensemble $\{\tau \in 2^{<\mathbb{N}} \ : \ \tau \Vdash^* \mathcal{R}\}$ est dense sous $\sigma$.*

Pour montrer le théorème, nous utiliserons deux lemmes. Le premier correspond à la proposition 4.6 pour la relation $\Vdash^*$.

**Lemme 4.15.** Supposons $\sigma \Vdash \mathcal{R}$ pour une chaîne $\sigma$ et un contrat $\mathcal{R}$. Si $\tau \succeq \sigma$, alors $\tau \Vdash \mathcal{R}$. ⋆

Preuve. Soit $\mathcal{B}_\mathcal{R}$ la classe des éléments qui satisfont $\mathcal{R}$. On a visisblement $[\tau] \setminus \mathcal{B}_\mathcal{R} \subseteq [\sigma] \setminus \mathcal{B}_\mathcal{R}$. Donc, si $[\sigma] \setminus \mathcal{B}_\mathcal{R}$ est maigre, alors $[\tau] \setminus \mathcal{B}_\mathcal{R}$ est maigre. ∎

**Lemme 4.16.** Supposons $\sigma \Vdash \mathcal{R}$ pour une chaîne $\sigma$ et un contrat $\mathcal{R}$. Alors, $\sigma \nVdash \neg\mathcal{R}$ ⋆

Preuve. Soit $\mathcal{B}_\mathcal{R}$ la classe des éléments qui satisfont $\mathcal{R}$, et soit $\mathcal{B}_{\neg\mathcal{R}}$ la classe des éléments qui satisfont $\neg\mathcal{R}$. Supposons par l'absurde $\sigma \Vdash \mathcal{R}$ et $\sigma \Vdash \neg\mathcal{R}$. Alors, les classes $[\sigma] \setminus (\mathcal{B}_\mathcal{R} \cap [\sigma])$ et $[\sigma] \setminus (\mathcal{B}_{\neg\mathcal{R}} \cap [\sigma])$ sont toutes deux maigres. Par ailleurs, $([\sigma] \setminus (\mathcal{B}_\mathcal{R} \cap [\sigma])) \ \cup \ ([\sigma] \setminus (\mathcal{B}_{\neg\mathcal{R}} \cap [\sigma])) = [\sigma]$, ce qui contredit le fait que la réunion de deux classes maigres soit maigre et donc d'intérieur vide. ∎

Preuve du théorème 4.14. Rappelons la proposition 4.8 : pour tout contrat $\mathcal{R}$ arithmétique et pour tout $\sigma$, si $\sigma \Vdash^* \mathcal{R}$, alors $\sigma \Vdash \mathcal{R}$.

Montrons que si $\{\tau \in 2^{<\mathbb{N}} \ : \ \tau \Vdash^* \mathcal{R}\}$ est dense sous $\sigma$, alors $\sigma \Vdash \mathcal{R}$. D'après la proposition 4.8, $A = \{\tau \in 2^{<\mathbb{N}} \ : \ \tau \Vdash \mathcal{R}\}$ est dense sous $\sigma$. La classe $\{X \in [\sigma] \ : \ \forall n \ X\upharpoonright_n \nVdash \mathcal{R}\}$ est donc un fermé d'intérieur vide, que l'on notera $\mathcal{F}$. Soit $\mathcal{B}_\mathcal{R}$ la classe des éléments qui satisfont $\mathcal{R}$. Par définition de $A$, pour tout $\tau \in A$, la classe $\mathcal{M}_\tau = [\tau] \setminus (\mathcal{B}_\mathcal{R} \cap [\tau])$ est maigre. Il s'ensuit que la classe $[\sigma] \setminus (\mathcal{B}_\mathcal{R} \cap [\sigma])$ est incluse dans la réunion de $\mathcal{F}$ et de toutes les classes $\mathcal{M}_\tau$. Elle est donc maigre.

Montrons enfin que si $\sigma \Vdash \mathcal{R}$, alors $\{\tau \in 2^{<\mathbb{N}} \ : \ \tau \Vdash^* \mathcal{R}\}$ est dense sous $\sigma$. Par contraposée, supposons qu'il existe $\tau \succeq \sigma$ tel que pour tout $\rho \succeq \tau$ on a $\rho \nVdash^* \mathcal{R}$. Au vu de la proposition 4.7, il doit alors exister $\rho \succeq \tau$ tel que $\rho \Vdash^* \neg\mathcal{R}$. Et, par la proposition 4.8, on a alors $\rho \Vdash \neg\mathcal{R}$. D'après le lemme 4.16, $\rho \nVdash \mathcal{R}$, et l'on a donc d'après le lemme 4.15 $\sigma \nVdash \mathcal{R}$. ∎

# 5. Ensembles arbitrairement génériques

La généricité peut être vue comme une notion de « typicité », au sens où tout ce qui peut arriver infiniment souvent finira par arriver. Dans le cas du forcing de Cohen, un ensemble dense a infiniment souvent la possibilité d'être rencontré, et cela arrivera si un ensemble est typique, ce qui correspond à la notion d'ensemble générique.

Nous avons vu les relativisations des ensembles faiblement 1-génériques et 1-génériques, et notamment les concepts de $n$-générique et faiblement $n$-générique. Nous étudions dans cette section les propriétés des ensembles typiques, c'est-à-dire les propriétés que vont avoir les ensembles suffisamment génériques.

Notons que si une propriété est vérifiée par tout ensemble suffisamment générique, elle est sans perte de généralité vérifiée pour tout ensemble faiblement 1-générique relativement à $A$ pour un certain oracle $A$ suffisamment puissant : il suffit que $A$ encode l'intersection d'ouverts denses correspondant au niveau de généricité requis. On s'attachera donc à déterminer « le bon niveau » de généricité nécessaire pour satisfaire telle ou telle propriété. En pratique, cela correspondra souvent à être $n$-générique pour un certain $n$.

## 5.1. Propriétés des ensembles suffisamment génériques

Nous avons vu que la 1-généricité était le niveau de généricité correspondant au forcing des contrats $\Sigma^0_1/\Pi^0_1$. Sans surprise, nous voyons à présent que la $n$-généricité est le niveau de généricité correspondant au forcing des contrats $\Sigma^0_n/\Pi^0_n$, et nous allons montrer le théorème suivant.

**Théorème 5.1**
*Soit $G$ un ensemble $n$-générique. Soit $\mathcal{R}$ un contrat $\Sigma^0_n$ ou $\Pi^0_n$. Alors, $G$ satisfait $\mathcal{R}$ si, et seulement si, il existe un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \mathcal{R}$.*

Notons que cette itération n'est pas triviale : il faut faire appel pour cela aux développements de la relation de forcing des sections précédentes.

Nous allons maintenant passer à la preuve du théorème 5.1 en exploitant la simplicité définitionnelle de la relation de forcing syntaxique. Commençons par la proposition suivante.

**Proposition 5.2.** Soit $\mathcal{R}$ un contrat arithmétique.

(1) Si $\mathcal{R}$ est $\Sigma^0_n$, alors le prédicat $\sigma \Vdash^* \mathcal{R}$ est $\Sigma^0_n$.

(2) Si $\mathcal{R}$ est $\Pi^0_n$, alors le prédicat $\sigma \Vdash^* \mathcal{R}$ est $\Pi^0_n$. ⋆

Preuve. Nous allons prouver (1) et (2) simultanément par induction sur la complexité arithmétique du contrat. Le cas $\Sigma^0_1$ et $\Pi^0_1$ a déjà été traité avec la proposition 3.17.

Si $\mathcal{R}$ est $\Sigma^0_{m+1}$ avec $m > 0$, alors il peut s'exprimer sous la forme $\exists x \mathcal{S}(x)$, où $\mathcal{S}$ est un contrat $\Pi^0_m$. On a $\sigma \Vdash^* \mathcal{R}$ si, et seulement si, il existe un $n \in \mathbb{N}$ tel $\sigma \Vdash^* \mathcal{S}(n)$. Par hypothèse d'induction, le prédicat $\sigma \Vdash^* \mathcal{S}(n)$ est $\Pi^0_m$, et le prédicat $\sigma \Vdash^* \mathcal{R}$ est donc $\Sigma^0_{m+1}$.

Si $\mathcal{R}$ est $\Pi^0_{m+1}$ avec $m > 0$, alors il peut s'exprimer sous la forme $\forall x \mathcal{S}(x)$ où $\mathcal{S}$ est un contrat $\Sigma^0_m$. On a $\sigma \Vdash^* \mathcal{R}$ ssi pour tout $\tau \succeq \sigma$ et tout $n$ dans $\mathbb{N}$, $\tau \nVdash^* \neg\mathcal{S}(n)$. En particulier, $\neg\mathcal{S}(n)$ est $\Pi^0_m$, donc par hypothèse d'induction, le prédicat $\tau \Vdash^* \neg\mathcal{S}(n)$ est $\Pi^0_m$, donc $\tau \nVdash^* \neg\mathcal{S}(n)$ est $\Sigma^0_m$, et le prédicat $\sigma \Vdash^* \mathcal{R}$ est $\Pi^0_{m+1}$. ∎

Nous voyons à présent le niveau de généricité requis pour rendre vraie la proposition 4.8.

**Proposition 5.3.** Soit $\mathcal{R}$ un contrat arithmétique tel que $\sigma \Vdash^* \mathcal{R}$ pour $\sigma$ dans $2^{<\mathbb{N}}$.

(1) Si $\mathcal{R}$ est $\Sigma^0_n$, alors $\mathcal{R}$ est satisfait pour tout faiblement $(n-2)$-générique qui étend $\sigma$.

(2) Si $\mathcal{R}$ est $\Pi^0_n$, alors $\mathcal{R}$ est satisfait pour tout faiblement $(n-1)$-générique qui étend $\sigma$. ⋆

Preuve. Par la définition de la relation de forcing, si $\mathcal{R}$ est $\Sigma^0_1$ ou $\Pi^0_1$, alors il est satisfait pour tout ensemble qui étend $\sigma$. Supposons la proposition vraie pour $n$. Soit $\mathcal{R} = \exists x \mathcal{Q}(x)$ un contrat $\Sigma^0_{n+1}$, avec $\mathcal{Q}(x)$ un contrat $\Pi^0_n$. On a $\sigma \Vdash^* \mathcal{R}$ si, et seulement si, il existe $m \in \mathbb{N}$ tel que $\sigma \Vdash^* \mathcal{Q}(m)$.

Par hypothèse d'induction, $\mathcal{Q}(m)$ est satisfait pour tout ensemble $(n-1)$-générique qui étend $\sigma$, et il en est donc de même pour $\mathcal{R}$.

Supposons à présent que $\mathcal{R} = \forall x \mathcal{Q}(x)$ est un contrat $\Pi^0_{n+1}$, avec $\mathcal{Q}(x)$ un contrat $\Sigma^0_n$. On a $\sigma \Vdash^* \mathcal{R}$ si, et seulement si, pour tout $m \in \mathbb{N}$ et pour tout $\tau \succeq \sigma$,

$$\tau \nVdash^* \neg Q(m).$$

D'après la proposition 4.7, cela signifie que pour $m$ entier fixé l'ensemble $A_m = \{\tau \ : \ \tau \Vdash^* Q(m)\}$ est dense sous $\sigma$. D'après la proposition 5.2, chaque ensemble $A_m$ est $\Sigma^0_n$. En particulier, si $G$ est faiblement $n$-générique et étend $\sigma$, alors $G$ rencontre $A_m$. Par hypothèse d'induction, si $G$ rencontre $A_m$ et est $n$-générique, alors $Q(m)$ est vrai pour $G$. Comme c'est le cas pour tout $m$, alors $\mathcal{R}$ est vrai pour tout ensemble $n$-générique $G$ qui étend $\sigma$. ∎

Nous pouvons finalement montrer le théorème 5.1.

Preuve du théorème 5.1. Le cas $\Sigma^0_1/\Pi^0_1$ a déjà été traité au moyen du théorème 3.12. Soit $n > 1$. On peut supposer sans perte de généralité que $\mathcal{R}$ est $\Sigma^0_n$. Dans le cas inverse, on reproduit l'argument suivant avec $\neg\mathcal{R}$ à la place de $\mathcal{R}$.

Soit $U = \{\sigma \in 2^{<\mathbb{N}} \ : \ \sigma \Vdash^* \mathcal{R}\}$. D'après le lemme 4.5,

$$U^\perp = \{\sigma \in 2^{<\mathbb{N}} \ : \ \sigma \Vdash^* \neg\mathcal{R}\}.$$

D'après la proposition 5.2, l'ensemble $U$ est $\Sigma^0_n$, et l'ensemble $U^\perp$ est $\Pi^0_n$. Notons que comme $G$ est $n$-générique, il rencontre $U \cup U^\perp$.

Supposons que $\mathcal{R}$ soit satisfait sur $G$. Alors, $G$ ne peut pas rencontrer $U^\perp$, car on aurait dans ce cas un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \neg\mathcal{R}$ et, par la proposition 5.3, $\neg\mathcal{R}$ serait donc satisfait sur $G$, ce qui contredit le fait que $\mathcal{R}$ soit satisfait sur $G$. Donc, $G$ rencontre $U$, et l'on a un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \mathcal{R}$.

Supposons que $\neg\mathcal{R}$ soit satisfait sur $G$. Symétriquement, $G$ rencontre nécessairement $U^\perp$, et l'on a un préfixe $\sigma \prec G$ tel que $\sigma \Vdash^* \mathcal{R}$.

Réciproquement, si un préfixe $\sigma \prec G$ est tel que $\sigma \Vdash^* \mathcal{R}$ ou $\sigma \Vdash^* \neg\mathcal{R}$, alors respectivement $\mathcal{R}$ ou $\neg\mathcal{R}$ est satisfait sur $G$, d'après la proposition 5.3. ∎

### 5.2. Degré Turing des ensembles suffisamment générique

Récapitulons d'abord un résultat que nous avons déjà établi : étant donné un ensemble $A$, si $G$ est suffisamment générique, alors $A$ ne calcule pas $G$ et $G$ ne calcule pas $A$.

**Théorème 5.4**
*Soit $A \in 2^{\mathbb{N}}$ un ensemble non calculable. Si $G$ est 1-générique relativement à $A$, alors $A$ et $G$ sont dans des degrés Turing incomparables.*

Preuve. D'après l'exercice 3.5 et l'exercice 3.6, si $G$ est faiblement 1-générique relativement à $A$, il est de degré $A$-hyperimmune, et ne peut donc être calculé par $A$. D'après le théorème 3.28, si $G$ est 1-générique relativement à $A$, il ne calcule pas $A$. ∎

Notons que le théorème 5.4 peut être généralisé pour montrer que pour toute suite dénombrable d'ensembles fixée à l'avance $A_0, A_1, \ldots$, tout ensemble $G$ suffisamment générique pour le forcing de Cohen est incomparable avec les éléments de cette liste. En effet, si $G$ est suffisamment générique, il sera 1-générique relativement à $A_i$ pour tout $i$. Ce résultat, malgré sa simplicité, admet plusieurs conséquences intéressantes.

**Corollaire 5.5**
*Soit $G$ un ensemble suffisamment générique pour le forcing de Cohen. Alors, $G$ n'est pas arithmétique et ne calcule aucun ensemble arithmétique non calculable.*

Voyons à présent comment renforcer et itérer le théorème 5.4 : pour $A$ non $\emptyset^{(n)}$-calculable, si un ensemble $G$ est suffisamment générique, non seulement il ne calculera pas $A$, mais aussi son $n$-ième saut Turing ne calculera pas $A$. Nous allons en fait voir quelque chose d'un peu plus précis : si $A$ n'est pas $\Sigma^0_n$ et si $G$ est suffisamment générique, alors $A$ ne sera pas $\Sigma^0_n(G)$. Assurons-nous d'abord de la complexité du contrat $a \in G^{(n)}$, via le lemme suivant.

**Lemme 5.6.** Pour tout $a \in \mathbb{N}$, le contrat $a \in X^{(n)}$ est $\Sigma^0_n$ uniformément en $a$. ⋆

Preuve. Pour le cas $n = 1$, on a $a \in X'$ ssi $\Phi_a(X, a)\downarrow$, ce qui est bien un contrat $\Sigma^0_1$. Au cas $n$, supposons que $F_n(G, x)$ soit une formule $\Sigma^0_n$ de l'arithmétique du second ordre telle que pour tout $X \in 2^{\mathbb{N}}$ et pour tout $a \in \mathbb{N}$, la formule $F(X, a)$ est vraie ssi $a \in X^{(n)}$. On a alors :

$$\begin{aligned} a \in X^{(n+1)} &\leftrightarrow \exists \sigma\ \forall i < |\sigma| \quad \left( \bigvee \begin{array}{l} (\sigma(i) = 0 \wedge i \notin X^{(n)}) \\ (\sigma(i) = 1 \wedge i \in X^{(n)}) \end{array} \right) \wedge \Phi_a(\sigma, a)\downarrow \\ &\leftrightarrow \exists \sigma\ \forall i < |\sigma| \quad \left( \bigvee \begin{array}{l} (\sigma(i) = 0 \wedge \neg F_n(X, i)) \\ (\sigma(i) = 1 \wedge F_n(X, i)) \end{array} \right) \wedge \Phi_a(\sigma, a)\downarrow. \end{aligned}$$

En utilisant le fait que $\neg F_n(G, x)$ et $F_n(G, x)$ sont toutes les deux des formules $\Sigma^0_{n+1}$, et en utilisant la clôture des formules $\Sigma^0_{n+1}$ par quantifi-

cation bornée, réunion finie et intersection finie, on peut définir une formule $\Sigma^0_{n+1}$ $H(G,\tau)$ telle que

$$H(X,\sigma) \leftrightarrow \forall i < |\sigma|\ ((\sigma(i) = 0 \wedge \neg F_n(X,i)) \vee (\sigma(i) = 1 \wedge F_n(X,i))).$$

Alors,

$$a \in X^{(n+1)} \leftrightarrow \exists \sigma\ H(X,\sigma)\ \wedge \Phi_a(\sigma,a)\downarrow,$$

ce qui est bien une formule $\Sigma^0_{n+1}$.

**Théorème 5.7**
*Soit $A$ un ensemble non $\Sigma^0_{n+1}$ pour un entier $n \geqslant 0$. Alors, pour tout ensemble $G$ 1-générique relativement à $A \oplus \emptyset^{(n)}$, $A$ n'est pas $\Sigma^0_{n+1}(G)$.*

Preuve. Soit $G$ un ensemble 1-générique relativement à $A \oplus \emptyset^{(n+1)}$. Il s'agit de montrer que pour tout $e$ l'ensemble $A$ est différent de $W_e^{G^{(n)}}$, l'ensemble $G^{(n)}$-c. e. de code $e$. Soit

$$U = \{\sigma\ :\ \exists m \notin A\ \sigma \Vdash^* m \in W_e^{G^{(n)}}\}.$$

D'après le lemme 5.6, le contrat $m \in W_e^{G^{(n)}}$ est $\Sigma^0_{n+1}$. Par suite, d'après la proposition 5.2, l'ensemble $U$ est $\Sigma^0_1(A \oplus \emptyset^{(n)})$. Si $G$ rencontre $U$, alors d'après la proposition 5.3 il existe $m \notin A$ tel que $m \in W_e^{G^{(n)}}$, et $A \neq W_e^{G^{(n)}}$. Si $G$ ne rencontre pas $U$, comme l'ensemble $G$ est 1-générique relativement à $A \oplus \emptyset^{(n+1)}$, $G$ rencontre $U^\perp = \{\sigma\ :\ \forall m \notin A\ \forall \tau \succeq \sigma\ \tau \nVdash^* m \in W_e^{G^{(n)}}\}$. D'après le lemme 4.5, on a

$$U^\perp = \{\sigma\ :\ \forall m\ \notin\ A\ \sigma \Vdash^* m\ \notin\ W_e^{G^{(n)}}\}.$$

Soit $\sigma \in U^\perp$ une chaîne fixée. Alors, l'ensemble

$$D_\sigma = \{m\ :\ \sigma \Vdash^* m \notin W_e^{G^{(n)}}\}$$

est un ensemble $\Pi^0_{n+1}$, qui par hypothèse contient $\mathbb{N} \setminus A$. Comme $\mathbb{N} \setminus A$ n'est pas $\Pi^0_{n+1}$, il y a forcément un élément $m \in A$ tel que $m \in D_\sigma$. Donc,

$$U^\perp \subseteq \{\sigma\ :\ \exists m \in A\ \sigma \Vdash^* m \notin W_e^{G^{(n)}}\}.$$

Alors, d'après la proposition 5.3, on aura $m\ \notin\ W_e^{G^{(n)}}$ pour $m \in A$, et donc $A \neq W_e^{G^{(n)}}$. ∎

**Corollaire 5.8**
*Soit $A$ un ensemble non $\emptyset^{(n)}$-calculable pour $n \geqslant 0$. Alors, si $G$ est 1-générique relativement à $A \oplus \emptyset^{(n)}$, l'ensemble $A$ n'est pas $G^{(n)}$-calculable.*

Preuve. Comme $A$ n'est pas $\emptyset^{(n)}$-calculable, il n'est pas $\Delta^0_{n+1}$. Comme $A$ n'est pas $\Delta^0_{n+1}$, soit $A$ n'est pas $\Sigma^0_{n+1}$, soit $\overline{A}$ n'est pas $\Sigma^0_{n+1}$. Par le

théorème 5.7, pour tout ensemble $G$ 1-générique relativement à $A \oplus \emptyset^{(n)}$, l'ensemble $A$ n'est pas $\Sigma^0_{n+1}(G)$ dans le premier cas, et $\overline{A}$ n'est pas $\Sigma^0_{n+1}(G)$ dans le second cas. Dans tous les cas, $A$ n'est pas $\Delta^0_{n+1}(G)$, et il n'est donc pas $G^{(n)}$-calculable. ∎

Nous avons vu avec le théorème 3.20 que le saut Turing était une fonction continue sur la classe des ensembles 1-génériques : les 1-génériques sont tous low généralisé. Ce résultat se relativise : le $n$-ième saut Turing est une fonction continue sur la classe des ensembles $n$-génériques.

**Théorème 5.9**
*Soit $G$ un ensemble $n$-générique. Alors, $G^{(n)} \leqslant_T G \oplus \emptyset^{(n)}$.*

Preuve. Soit $n > 0$, et soit $G$ un ensemble $n$-générique. Soit

$$U_e = \{\sigma \ : \ \sigma \Vdash^* e \in G^{(n)}\}.$$

D'après le lemme 5.6, le contrat $e \in G^{(n)}$ est $\Sigma^0_n$, et $U_e$ est donc un ensemble $\Sigma^0_n$. Considérons $U_e^\perp = \{\sigma \ : \ \forall \tau \succeq \sigma \ \tau \nVdash^* e \in G^{(n)}\}$. En particulier, $U_e^\perp$ est un ensemble $\Pi^0_n$. D'après le lemme 4.5,

$$U_e^\perp = \{\sigma \ : \ \sigma \Vdash^* e \notin G^{(n)}\}.$$

Comme $G$ est $n$-générique, il rencontre $U_e \cup U_e^\perp$. Il suffit à l'aide de $\emptyset^{(n)}$ de chercher un préfixe de $G$ dans $U_e$ ou $U_e^\perp$. D'après la proposition 5.3, on a dans le premier cas $e \in G^{(n)}$, et dans le second $e \notin G^{(n)}$. ∎

Notons que le corollaire 5.8 se déduit aussi du théorème précédent et d'une version relativisée du théorème 3.28. Attention ! Dans la littérature, la notion de low$_n$-généralisé ne correspond pas à la propriété du théorème 5.9.

**Définition 5.10.** Un ensemble $G \in 2^{\mathbb{N}}$ est *low$_n$ généralisé* si

$$G^{(n)} \leqslant_T (G \oplus \emptyset')^{(n-1)}.$$ ♢

Notons que si $X$ est low$_n$ généralisé, il est aussi low$_{n+1}$ généralisé. Chaque ensemble $n$-générique est bien low$_n$ généralisé, mais le théorème 5.9 prouve quelque chose de plus fort.

Chapitre 11

# Forcing effectif

Fort des intuitions créées avec l'étude du forcing de Cohen, nous pouvons alors introduire les notions abstraites de forcing sur un ordre partiel arbitraire, et développer toute la machinerie associée.

## 1. Fondements du forcing

Maintenant que nous nous sommes familiarisés avec les notions de densité et de généricité sur l'ordre partiel des chaînes binaires, muni de la relation d'extension, nous sommes prêts à aborder les concepts du forcing dans toute leur généralité, tout en conservant les intuitions de la méthode des extensions finies. Les bénéfices de cette abstraction n'apparaîtront qu'à partir de l'introduction de la relation de forcing, qui donne toute sa puissance au formalisme. Jusqu'ici, nous avons vu la notion de généricité dans l'ordre partiel des chaînes binaires comme une systématisation des constructions avec la méthode des extensions finies.

**Ordre partiel.** Dans toute sa généralité, une notion de forcing est tout simplement un ordre partiel $(\mathbb{P}, \leqslant)$, dont les éléments sont appelés *conditions*. Une condition représente intuitivement une approximation de l'objet que l'on est en train de construire. Une *extension* de $c \in \mathbb{P}$ est une condition $d \leqslant c$.

**Remarque**

Attention, pour des raisons historiques, la relation d'ordre du forcing est inversée. Une extension d'une condition $c$ est donc une condition $d$ plus petite. L'idée sous-jacente vient du fait que $d$ est une approximation plus précise que $c$, et qu'ainsi l'ensemble des objets « candidats » que l'on

est en train de construire est un sous-ensemble des candidats de $c$, car plus l'on ajoute de contraintes, plus l'on exclut des candidats.

**Filtre.** Dans le cas de la méthode des extensions finies, l'objet construit est un élément de $2^{\mathbb{N}}$, à l'aide d'une suite infinie strictement croissante de chaînes. Dans le langage d'un ordre partiel arbitraire, nous allons construire une suite infinie décroissante de conditions. L'objet produit est un filtre maximal.

**Définition 1.1.** Deux conditions $c_0, c_1$ sont *compatibles* s'il existe une condition $d$ qui étend à la fois $c_0$ et $c_1$. Dans le cas contraire, $c_0$ et $c_1$ sont *incompatibles*. Un *filtre* est un ensemble $F \subseteq \mathbb{P}$ clos par le haut, tel que pour tous $c_0, c_1 \in F$, il existe une condition $d \in F$ telle que $d \leqslant c_0, c_1$. Un filtre est *maximal* s'il n'est pas inclus dans un filtre strictement plus grand. ♢

Si l'on considère l'ordre partiel sur les chaînes, muni de la relation de suffixe, les filtres maximaux sont en correspondance bijective avec l'espace de Cantor. En effet, pour tout $X \in 2^{\mathbb{N}}$, l'ensemble $\{X \restriction_n : n \in \mathbb{N}\}$ est un filtre maximal, et tout filtre maximal est inversement de cette forme.

Dans le cadre d'un ordre partiel dénombrable, il peut être plus intuitif de se représenter un filtre comme la clôture par le haut d'une suite infinie décroissante de conditions. En particulier, pour toute chaîne infinie décroissante de conditions $c_0 \geqslant c_1 \geqslant c_2 \geqslant \dots$, l'ensemble

$$F = \{d \in \mathbb{P} : \exists n \; c_n \leqslant d\}$$

est un filtre. Cette intuition correspond plus à la construction de la méthode des extensions finies.

**Notation**

Comme expliqué, une condition $c \in \mathbb{P}$ peut être vue comme une approximation de l'objet que l'on construit, à savoir un filtre maximal. On peut donc associer à chaque condition l'ensemble $[c]_\in$ des filtres maximaux contenant $c$, représentant les objets candidats. En particulier, suivant l'intuition, si $d \leqslant c$, alors l'approximation $d$ est plus précise que l'approximation $c$, réduisant ainsi le nombre de candidats. On a dont $[d]_\in \subseteq [c]_\in$.

Remarquons que pour tout filtre maximal $F$, $\bigcap_{c \in F} [c]_\in = \{F\}$. Autrement dit, $F$ est l'unique candidat de toutes les conditions du filtre simultanément. Nous avons déjà rencontré plusieurs notions de forcing au cours des chapitres précédents. En voici quelques-uns. Nous allons voir que pour chacune de ces notions de forcing, les filtres maximaux peuvent être interprétés comme des ensembles d'entiers.

**Exemple 1.2.**

1. Le *forcing de Cohen* est l'ordre partiel des chaînes muni de sa relation de suffixe $(2^{<\mathbb{N}}, \succeq)$. Pour tout $\sigma \in 2^{<\mathbb{N}}$, $[\sigma]_\in$ est en bijection avec l'ensemble $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \prec X\}$, par la fonction qui à $F \in [\sigma]_\in$ associe l'unique élément de $\bigcap_{\sigma \in F} [\sigma]$.
2. Le *forcing de Jockusch–Soare* est l'ordre partiel des classes $\Pi^0_1$ non vides, ordonnés par la relation d'inclusion. Pour toute classe $\Pi^0_1$ $\mathcal{P}$, l'ensemble $[\mathcal{P}]_\in$ est en bijection avec $\mathcal{P}$. Un filtre maximal $F$ contenant $\mathcal{P}$ peut donc être vu comme l'unique élément de $\bigcap_{\mathcal{Q} \in F} \mathcal{Q}$, qui est un membre de $\mathcal{P}$.
3. Le *forcing de Sacks* est l'ordre partiel des $f$-arbres calculables, ordonnées par la relation de sous-$f$-arbre. Pour tout $f$-arbre calculable $T$, l'ensemble $[T]_\in$ est en bijection avec l'ensemble

$$[T] = \Big\{ \bigcup_n T(X\restriction_n) : X \in 2^{\mathbb{N}} \Big\} = \{Y \in 2^{\mathbb{N}} : \exists^\infty n \; Y\restriction_n \; \in \operatorname{Im} T\}.$$

Dans chacun des exemples précédents, pour tout filtre maximal $F$, on notera $\dot{F}$ l'élément de $2^{\mathbb{N}}$ correspondant. On a donc pour chacune de ces notions de forcing l'égalité $[c] = \{\dot{F} : F \in [c]_\in\}$.

**Densité, généricité.** Rappelons que dans la méthode des extensions finies, les contrats peuvent être représentés comme l'ensemble des chaînes qui les forcent. Cette représentation a l'avantage de s'abstraire de la notion de contrat et se généralise à tout ordre partiel.

**Définition 1.3.** Soit $(\mathbb{P}, \leqslant)$ un ordre partiel. Un ensemble $D \subseteq \mathbb{P}$ est *dense* dans $(\mathbb{P}, \leqslant)$ si, pour tout $c \in \mathbb{P}$, il existe un $d \leqslant c$ tel que $d \in D$. ♢

L'intuition qu'il est utile de conserver avec la notion de densité est que si un ensemble $D$ est dense, alors quel que soit le morceau fini de la suite décroissante de conditions que l'on a déjà construit, il n'est jamais trop tard pour intégrer un élément de $D$ dans la suite.

**Définition 1.4.** Soit $\vec{D} = (D_n)_{n \in \mathbb{N}}$ une collection d'ensembles de conditions. Un filtre $F \subseteq \mathbb{P}$ est $\vec{D}$-*générique* s'il intersecte l'ensemble $D_n$ pour tout $n \in \mathbb{N}$. ♢

La proposition suivante montre que si les ensembles de conditions sont denses, il existe un filtre générique pour ces ensembles. La preuve de cette proposition correspond à la construction par la méthode des extensions finies d'une suite infinie croissante de chaînes satisfaisant chaque contrat.

**Proposition 1.5.** Soit $\vec{D} = (D_n)_{n\in\mathbb{N}}$ une collection d'ensembles denses et soit $c \in \mathbb{P}$ une condition. Il existe un filtre $\vec{D}$-générique contenant $c$. ⋆

Preuve. Définissons inductivement une suite infinie décroissante de conditions $c_0 \geqslant c_1 \geqslant c_2 \geqslant \dots$ comme suit : $c_0 = c$. Si $c_n$ est défini, $c_{n+1} \leqslant c_n$ est une condition appartenant à $D_n$. Une telle extension de $c_n$ existe par densité de l'ensemble $D_n$. Soit $F = \{d \in \mathbb{P} : \exists n\ d \geqslant c_n\}$. L'ensemble $F$ est un filtre contenant $c$ et rencontrant $D_n$ pour tout $n$. ■

Comme expliqué dans la section 10-2, il existe en général une quantité indénombrable d'ensembles denses, et un filtre $F$ ne peut pas être générique pour tous ces ensembles simultanément. La notion de généricité est donc dépendante d'une collection $\vec{D}$ dénombrable d'ensembles denses. Nous dirons que tout filtre *suffisamment générique* pour une notion de forcing satisfait telle propriété s'il existe une collection dénombrable d'ensembles denses $\vec{D}$ telle que tout filtre $\vec{D}$-générique satisfait cette propriété.

**Définition 1.6.** Soit $(\mathbb{P}, \leqslant)$ un ordre partiel, et soient $c \in \mathbb{P}$ et $D \subseteq \mathbb{P}$ un ensemble. On dit que $D$ est *dense sous* $c$ si pour tout $d \leqslant c$, il existe un $e \leqslant d$ tel que $e \in D$. ♢

L'exercice suivant sera utile dans la suite de ce développement.

**Exercice 1.7.** Supposons qu'un ensemble $D \subseteq \mathbb{P}$ est dense sous $c \in \mathbb{P}$. Montrer que tout filtre $F$ suffisamment générique contenant la condition $c$ intersecte $D$. ◇

## 2. Relation de forcing

Jusqu'ici, nous avons développé des notions s'exprimant purement en termes d'ordre partiel, à savoir, les notions de densité, de filtre et de généricité. Nous allons maintenant définir une généralisation de la relation de forcing introduite dans la section 10-4. Pour cela, nous allons nous restreindre à des notions de forcing produisant des ensembles d'entiers.

**Définition 2.1.** Un *forcing de Cantor* est un ordre partiel $(\mathbb{P}, \leqslant)$ muni d'une fonction $F \mapsto \dot{F}$ des filtres maximaux vers l'espace de Cantor $2^{\mathbb{N}}$, telle pour tout $c \in \mathbb{P}$, et tout $\sigma \in 2^{<\mathbb{N}}$ pour lequel $[c] \cap [\sigma] \neq \emptyset$, il existe une condition $d \leqslant c$ telle que $[d] \subseteq [\sigma]$. Ici, la notation $[c]$ désigne l'ensemble $\{\dot{F} : F \in [c]_\in\}$. ♢

### 2.1. Relation sémantique de forcing

Par abus de langage, on dira qu'un ensemble $G \in 2^{\mathbb{N}}$ est *suffisamment générique* pour un forcing de Cantor s'il est de la forme $\dot{F}$ pour un filtre suffisamment générique.

**Définition 2.2.** Une condition $c$ *force sémantiquement* un contrat $\mathcal{R}$, auquel cas l'on note $c \Vdash \mathcal{R}$ si $\dot{F}$ satisfait $\mathcal{R}$ pour tout filtre maximal $F$ suffisamment générique et contenant $c$. ♢

Cette définition sémantique nous donne « gratuitement » certaines propriétés de la relation.

**Proposition 2.3.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et soient $c, d \in \mathbb{P}$. Soit $\mathcal{R}$ un contrat.

(1) Si $c \Vdash \mathcal{R}$ et $d \leqslant c$, alors $d \Vdash \mathcal{R}$.

(2) Si $c \Vdash \mathcal{R}$, alors $c \nVdash \neg\mathcal{R}$. ⋆

Preuve. (1) Soit $F$ un filtre suffisamment générique contenant $d$. Par clôture par le haut des filtres, $c \in F$, et comme $c$ force $\mathcal{R}$, $\dot{F}$ satisfait donc $\mathcal{R}$. Il s'ensuit que $d$ force $\mathcal{R}$.

(2) Si $c$ force $\mathcal{R}$ et $\neg\mathcal{R}$, alors pour tout filtre $F$ suffisamment générique contenant $c$, $\dot{F}$ satisfait $\mathcal{R}$ et $\neg\mathcal{R}$, contradiction. ■

**Exercice 2.4.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et soit $c \in \mathbb{P}$. Soit $\mathcal{R}$ un contrat. Montrer que si $\{d \in \mathbb{P} : d \Vdash \mathcal{R}\}$ est dense sous $c$, alors $c \Vdash \mathcal{R}$. ⋄

En revanche, certaines propriétés sont beaucoup moins évidentes à prouver. Nous allons voir par exemple que pour tout contrat arithmétique $\mathcal{R}$, l'ensemble des conditions qui forcent $\mathcal{R}$ ou qui forcent $\neg\mathcal{R}$ est dense. Tout comme dans le cas du forcing de Cohen avec l'ordre partiel des chaînes $(2^{<\mathbb{N}}, \succeq)$, nous allons définir une relation de forcing syntaxique qui nous permettra de rendre compte de ce phénomène.

Nous insistons auparavant sur les trois propriétés fondamentales que doit avoir une relation de forcing syntaxique.

**Définition 2.5 (Relation de forcing).** Étant donné $(\mathbb{P}, \leqslant)$ un forcing de Cantor, on appelle *relation de forcing* une relation $\Vdash^\circ$ satisfaisant les propriétés suivantes pour tous $c, d \in \mathbb{P}$ et tout contrat arithmétique $\mathcal{R}$.

(1) Si $c \Vdash^\circ \mathcal{R}$, alors $c \Vdash \mathcal{R}$.

(2) Si $c \Vdash^\circ \mathcal{R}$ et $d \leqslant c$, alors $d \Vdash^\circ \mathcal{R}$.

(3) L'ensemble $\{c \in \mathbb{P} : c \Vdash^\circ \mathcal{R} \text{ ou } c \Vdash^\circ \neg\mathcal{R}\}$ est dense. ♢

Les propriétés (1-3) correspondent aux propositions 10-4.8, 10-4.6 et 10-4.7 pour le forcing de Cohen. Il découle immédiatement de (1) que si $c \Vdash^\circ \mathcal{R}$, alors $c \nVdash^\circ \neg\mathcal{R}$. Rappelons qu'un ensemble $S \subseteq \mathbb{P}$ est dense sous $c \in \mathbb{P}$ si pour tout $d \leqslant c$, il existe un $e \leqslant d$ tel que $e \in S$.

**Proposition 2.6.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et soit $c \in \mathbb{P}$. Soient $\mathcal{R}$ un contrat arithmétique et $\Vdash^\circ$ une relation de forcing. Alors,

$$c \Vdash \mathcal{R} \quad \text{si, et seulement si,} \quad \{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$$

est dense sous $c$. ⋆

Preuve. Supposons que $c \Vdash^\circ \mathcal{R}$. Soit $d \leqslant c$. Par la définition 2.5(3), il existe un $e \leqslant c$ tel que $e \Vdash^\circ \mathcal{R}$ ou $e \Vdash^\circ \neg\mathcal{R}$. Si le second cas se présente, alors par la définition 2.5(1), $e \Vdash \neg\mathcal{R}$. On a donc $e \Vdash \mathcal{R}$ et $e \Vdash \neg\mathcal{R}$, ce qui contredit la proposition 2.3(2). Le premier cas se présente donc. Nous avons montré la densité de l'ensemble $\{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$ sous $c$.

Réciproquement, supposons que l'ensemble $\{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$ soit dense sous $c$. Soit $F$ un filtre suffisamment générique contenant $c$. Par généricité, il existe $d \in F$ tel que $d \Vdash^\circ \mathcal{R}$, et par la définition 2.5(1), $d \Vdash \mathcal{R}$, donc $\dot{F}$ satisfait $\mathcal{R}$. Il s'ensuit que $c \Vdash \mathcal{R}$. ∎

### 2.2. Relation syntaxique de forcing

Nous définissons à présent notre relation syntaxique de forcing $\Vdash^*$, qui généralise directement la relation $\Vdash^*$ définie dans la section 10-4 pour le forcing de Cohen. Tout comme pour le forcing de Cohen, l'intérêt de la relation $c \Vdash^* \mathcal{R}$ est qu'elle est simple à définir : relativement à $\mathbb{P}$, elle a la même complexité arithmétique que celle du contrat $\mathcal{R}$. Nous verrons dans les sections suivantes que $\mathbb{P}$ en tant qu'ordre partiel est malheureusement rarement calculable, ce qui amène à des complexités calculatoires supplémentaires qui doivent être traitées au cas par cas.

**Définition 2.7.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor. On définit la relation $\Vdash^*$ pour tout $c \in \mathbb{P}$ et tout contrat arithmétique $\mathcal{R}$ :

(1) $c \Vdash^* \mathcal{R}$ pour $\mathcal{R}$ un contrat $\Sigma^0_1$ ou $\Pi^0_1$ ssi tout $X \in [c]$ satisfait $\mathcal{R}$ ;

(2) $c \Vdash^* \exists x \mathcal{R}(x)$ pour $\mathcal{R}(x)$ un contrat $\Pi^0_k$ avec $k \geqslant 1$ ssi il existe un $n \in \mathbb{N}$ tel que $c \Vdash^* \mathcal{R}(n)$ ;

(3) $c \Vdash^* \forall x \mathcal{R}(x)$ pour $\mathcal{R}(x)$ un contrat $\Sigma^0_k$ avec $k \geqslant 1$ ssi pour tout $d \leqslant c$ et tout $n \in \mathbb{N}$, $d \nVdash^* \neg\mathcal{R}(n)$. ♢

Notons que tout comme dans le cas du forcing de Cohen, on a $c \Vdash^* \forall x \mathcal{R}(x)$ ssi $\forall d \leqslant c\ d \nVdash^* \exists x \neg\mathcal{R}(x)$. Nous laissons en exercice la preuve que la relation $\Vdash^*$ respecte les points (1-3) d'une relation de forcing de la définition 2.5. Il s'agit à chaque fois de simples preuves par induction sur la complexité des contrats.

**Exercice 2.8.** $(\star)$ Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et soient $c, d \in \mathbb{P}$. Soit $\mathcal{R}$ un contrat arithmétique. Montrer que si $c \Vdash^* \mathcal{R}$ et $d \leqslant c$, alors $d \Vdash^* \mathcal{R}$. ◇

**Exercice 2.9.** $(\star)$ Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor et soit $\mathcal{R}$ un contrat arithmétique. Montrer que $\{c \in \mathbb{P} : c \Vdash^* \mathcal{R} \text{ ou } c \Vdash^* \neg\mathcal{R}\}$ est dense. ◇

**Exercice 2.10.** $(\star)$ Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et soit $c \in \mathbb{P}$. Soit $\mathcal{R}$ un contrat arithmétique. Montrer que si $c \Vdash^* \mathcal{R}$, alors $c \Vdash \mathcal{R}$. ◇

**La complexité de $\Vdash$**

La complexité de $\Vdash^*$ a des conséquences sur la complexité de la relation sémantique de forcing $\Vdash$. D'après les exercices précédents, $\Vdash^*$ respecte bien les points (1-3) de la définition 2.5. Donc, d'après la proposition 2.6, la relation $c \Vdash \mathcal{R}$ est équivalente à $\forall d \leqslant c \; \exists e \leqslant d \; e \Vdash^* \mathcal{R}$, ce qui par exemple pour un contrat $\Sigma^0_n$ sera un prédicat $\Pi^0_{n+1}(\mathbb{P})$. Il s'agit d'une simplification considérable de la relation sémantique, mais qui reste — relativement à $\mathbb{P}$ — de complexité arithmétique supérieure à celle des contrats qu'elle force, ce qui nous fera préférer la relation $\Vdash^*$.

# 3. Forcing avec des arbres

Le forcing à base d'arbres est une des grandes familles de forcing. Nous en détaillons ici deux exemples, déjà rencontrés dans les chapitres précédents : le forcing de Jockusch-Soare et le forcing de Sacks calculable. Ces notions ont été au départ créées pour contrôler le simple saut via le forcing de contrats $\Sigma^0_1/\Pi^0_1$, ou le double saut via le forcing de contrats $\Sigma^0_2/\Pi^0_2$. Nous en rediscuterons dans la section 4, et nous nous contentons pour le moment de voir en quoi ces forcings ont été utilisés implicitement à plusieurs reprises dans ce livre.

## 3.1. Forcing de Jockusch-Soare

Le forcing de Jockusch-Soare correspond à l'ordre partiel des classes $\Pi^0_1$ non vides, partiellement ordonnées par la relation d'inclusion. On peut associer à tout filtre maximal $F$ sur cet ordre un ensemble d'entiers $\dot{F} \in 2^{\mathbb{N}}$ qui est l'élément du singleton $\bigcap_{\mathcal{P} \in F} \mathcal{P}$. Muni de cette interprétation des filtres, le forcing de Jockusch-Soare est un forcing de Cantor (voir la définition 2.1).

Nous avons déjà rencontré plusieurs utilisations du forcing de Jockusch-Soare dans le chapitre 8 sur les classes $\Pi^0_1$ et les degrés PA. Voici une reformulation des théorèmes de base calculatoirement dominée et de base d'évitement de cône, via le vocabulaire du forcing.

**Théorème (reformulation des th. 8-4.5 et 8-4.7)**
*Soit $A$ un ensemble non calculable et soit $G$ un ensemble suffisamment générique pour le forcing de Jockusch-Soare. Alors, $G$ est calculatoirement dominé et ne calcule pas $A$.*

Preuve. Soit $(\mathbb{P}, \leqslant)$ le forcing Jockusch-Soare, c'est-à-dire l'ensemble des classes $\Pi^0_1$ non vides ordonnées pas l'inclusion. La preuve du théorème 8-4.5 montre la chose suivante : pour toute fonctionnelle $\Phi_e$ et tout $c \in \mathbb{P}$, il existe $d \leqslant c$ tel que $d \Vdash^* \exists n\ \Phi_e(G, n)\!\uparrow$ ou il existe $d \leqslant c$ et une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ telle que $d \Vdash^* \Phi_e(G, n)\!\downarrow < f(n)$ pour tout $n$. Donc, l'ensemble $C_e$ des conditions qui forcent $\Phi_e$ à être partielle ou bien bornée par une fonction calculable est dense. Tout filtre suffisamment générique contient une condition dans chacun des $C_e$. Si $G \in 2^{\mathbb{N}}$ est donc suffisamment générique, il est calculatoirement dominé.

La preuve du théorème 8-4.7 quant à elle montre la chose suivante : pour toute fonctionnelle $\Phi_e$ l'ensemble $D_e = \{c \in \mathbb{P} \ :\ c \Vdash^* \exists n\ \Phi_e(G, n)\!\uparrow$ ou $\exists n\ c \Vdash^* \Phi_e(G, n)\!\downarrow \neq A(n)\}$ est dense. Si $G$ est donc suffisamment générique, il ne calcule pas $A$. ∎

En ce qui concerne le théorème de la base low, les choses sont différentes : c'est bien le forcing de Jockusch-Soare que nous utilisons pour construire un ensemble low, mais il s'agit d'une utilisation effective de ce forcing, où l'on contrôle à l'aide de $\emptyset'$ le passage d'une étape $n$ à une étape $n+1$. Il ne s'agit donc pas à proprement parler d'un résultat de forcing, dans le sens où ce n'est pas une propriété satisfaite par tout ensemble suffisamment générique, mais au contraire par une petite classe dénombrable d'ensembles qui sont peu génériques.

Nous avons vu avec le forcing de Cohen que tout ensemble suffisamment générique différait d'une quantité dénombrable d'ensembles fixés à l'avance (voir le théorème 10-5.4). En particulier, aucun ensemble suffisamment générique pour le forcing de Cohen n'est arithmétique. Ce n'est pas le cas du forcing de Jockusch-Soare. En effet, pour tout $X$ calculable, le singleton $\{X\}$ est une classe $\Pi^0_1$ et l'unique filtre contenant $\{X\}$ — à savoir le filtre de toutes les classes $\Pi^0_1$ ayant $X$ comme chemin infini — est maximal. L'ensemble $X$ est donc aussi générique que l'on veut sous la condition $\{X\}$.

Il est toutefois possible de modifier légèrement le forcing de Jockusch-Soare afin d'éviter les éléments calculables.

**Proposition 3.2.** Soit $(\mathbb{P}, \leqslant)$ l'ordre partiel des classes $\Pi^0_1$ non vides sans élément calculable, ordonnées par l'inclusion. Soit $(A_n)_{n\in\mathbb{N}}$ une suite quelconque d'ensembles. Alors, si $G \in 2^{\mathbb{N}}$ est suffisamment générique pour $\mathbb{P}$, il est différent de chaque $A_n$. ⋆

Preuve. Fixons un ensemble $A$ quelconque et montrons que si $G$ est suffisamment générique, il est différent de $A$. Le résultat suivra automatiquement pour la suite $(A_n)_{n\in\mathbb{N}}$.

Soit $D \subseteq \mathbb{P}$ l'ensemble des classes $\Pi^0_1$ de $\mathbb{P}$ ne contenant pas $A$. Montrons que $D$ est dense dans $\mathbb{P}$. Soit $\mathcal{P} \in \mathbb{P}$. Par la proposition 8-3.6, la classe $\mathcal{P}$ contient au moins deux éléments. Soit $B \in \mathcal{P}$ tel que $B \neq A$, et soit $n \in \mathbb{N}$ tel que $A(n) \neq B(n)$. Alors, la classe $\mathcal{Q} = \{X \in \mathcal{P} : X(n) = B(n)\}$ est une classe $\Pi^0_1$ non vide incluse dans $\mathcal{P}$ et telle que $\mathcal{Q} \in D$. Donc, $D$ est dense.■

La modification du forcing de Jockusch-Soare de la proposition précédente peut se décliner de multiples manières pour obtenir différents résultats : on peut considérer l'ordre partiel des classes $\Pi^0_1$ non vides ne contenant que des degrés PA, ou encore celles des classes $\Pi^0_1$ non vides ne contenant que des ensembles aléatoires au sens de Martin-Löf (voir le chapitre 18).

### 3.2. Le forcing de Sacks calculable

Le forcing de Sacks désigne de manière générale l'ordre partiel des arbres parfaits de $2^{<\mathbb{N}}$, sans restriction particulière d'effectivité. Nous nous restreignons en calculabilité aux arbres parfaits calculables, qui ont déjà été abordés via la notion de *f-arbre*.

Rappelons qu'un *f-arbre* est une fonction totale $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ telle que pour tous $\sigma, \tau \in \operatorname{dom} T$, $\sigma \preceq \tau$ si, et seulement si, $T(\sigma) \preceq T(\tau)$. Un *sous-f-arbre* d'un f-arbre $T$ est un f-arbre $S$ tel que $\operatorname{Im} S \subseteq \operatorname{Im} T$. Un f-arbre $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ s'étend en une fonction $\widehat{T} : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ définie par $\{\widehat{T}(X)\} = \bigcap_n [T(X \upharpoonright_n)]$. Un *chemin* de $T$ est un élément de $\operatorname{Im} \widehat{T}$. On note $[T]$ l'ensemble des chemins de $T$.

On appelle *forcing de Sacks calculable* l'ensemble des f-arbres calculables partiellement ordonnés par la relation de sous f-arbre. Comme il n'y aura pas d'ambiguïté possible dans ce livre, on dira parfois plus simplement *forcing de Sacks*. On peut associer à tout filtre maximal $F$ sur cet ordre un ensemble d'entiers $\dot{F} \in 2^{\mathbb{N}}$ qui est l'élément du singleton $\bigcap_{T\in F} [T]$. Muni de cette interprétation des filtres, le forcing de Sacks est un forcing de Cantor (voir la définition 2.1).

**Remarque**

Rappelons qu'une classe $\mathcal{P} \subseteq 2^{\mathbb{N}}$ est parfaite si $\mathcal{P} = [T]$ pour un f-arbre $T$. Le forcing de Sacks n'est cependant pas la restriction du forcing de Jockusch-Soare aux classes $\Pi^0_1$ parfaites : certaines classes $\Pi^0_1$ parfaites ne peuvent pas nécessairement être représentées par un f-arbre calculable.

En effet, pour tout f-arbre calculable $T$, $[T]$ contient une infinité d'éléments calculables, à savoir $T(X)$ pour tout ensemble calculable $X$, ce qui n'est par exemple pas le cas de la classe $\Pi^0_1$ des fonctions $\mathrm{DNC}_2$, qui est pourtant bien une classe parfaite.

Comme expliqué dans la remarque précédente, tout f-arbre calculable possède une infinité de chemins calculables. En revanche, tout ensemble suffisamment générique pour le forcing de Sacks sera non calculable.

**Exercice 3.3. (⋆)** Soit $(A_n)_{n\in\mathbb{N}}$ une suite quelconque d'ensembles. Montrer que tout $G$ suffisamment générique pour le forcing de Sacks est différent de chaque $A_n$. ◇

Un examen attentif de la première preuve que nous avons donnée de l'existence d'un degré calculatoirement dominé différent de $\mathbf{0}$, à l'aide de f-arbre, montre en fait que l'on a le résultat suivant.

**Théorème (reformulation du théorème 7-5.6)**
*Soit $G$ un ensemble suffisamment générique pour le forcing de Sacks. Alors, $G$ est non calculable et calculatoirement dominé.*

Preuve. Soit $(\mathbb{P}, \leqslant)$ le forcing de Sacks. La preuve du théorème 7-5.6 montre la chose suivante : pour toute fonctionnelle $\Phi_e$ et tout $c \in \mathbb{P}$, il existe $d \leqslant c$ tel que $d \Vdash^* \exists n\ \Phi_e(G, n)\!\uparrow$ ou il existe $d \leqslant c$ et une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ telle que $d \Vdash^* \Phi_e(G, n)\!\downarrow < f(n)$ pour tout $n$. Donc, l'ensemble $C_e$ des conditions qui forcent $\Phi_e$ à être partielle ou bien bornée par une fonction calculable est dense. Tout filtre suffisamment générique contient une condition de chacun des $C_e$. Donc si $G \in 2^{\mathbb{N}}$ est suffisamment générique il est calculatoirement dominé.

Par ailleurs, l'exercice 3.3 montre que si $G$ est suffisamment générique, il est non calculable. ∎

**Exercice 3.5. (⋆)** Soit $A$ un ensemble non calculable. Montrer que si $G$ est suffisamment générique pour le forcing de Sacks, il ne calcule pas $A$. ◇

**Exercice 3.6. (⋆⋆)** Un ensemble $X$ est *calculatoirement traçable* (Terwijn et Zambella [222]) s'il existe une borne calculable $h : \mathbb{N} \to \mathbb{N}$ telle que pour toute fonction $X$-calculable $f : \mathbb{N} \to \mathbb{N}$, il existe une suite calculable $(T_n)_{n\in\mathbb{N}}$ d'ensembles finis tels que $|T_n| \leqslant h(n)$ et tels que $f(n) \in T_n$ pour tout $n$.

1. Montrer que si $X$ est suffisamment générique pour le forcing de Sacks, il est calculatoirement traçable.

2. Montrer que si $X$ est calculatoirement traçable via une borne calculable $h$, alors il est calculatoirement traçable pour n'importe quelle borne calculable $h'$ telle que $h'(n) \leqslant h'(n+1)$ et $\lim_n h(n) = +\infty$ (Terwijn et Zambella [222]).
3. Soit $(2^n)^{\mathbb{N}}$ l'ensemble des fonctions $f : \mathbb{N} \to \mathbb{N}$ telles que $f(n) < 2^n$, et soit $(2^n)^{<\mathbb{N}}$ l'ensemble des préfixes de fonctions de $(2^n)^{\mathbb{N}}$. Soit $(\mathbb{P}, \leqslant)$ l'ensemble des conditions de forcing données par $T \in \mathbb{P}$ si $T \subseteq (2^n)^{<\mathbb{N}}$ est un arbre calculable qui vérifie la propriété suivante :
pour tout $\sigma \in T$, il existe $\tau \in T$ avec $\tau \succeq \sigma$ telle que pour tout $i < 2^{|\tau|}$ la chaîne $\tau i$ est dans $T$ ; autrement dit, chaque nœud a une extension de branchement maximal.
Montrer que tout ensemble suffisamment générique pour ce forcing est calculatoirement dominé et non calculatoirement traçable. ◇

# 4. Complexité calculatoire et question de forcing

Le forcing de Cohen présente une particularité qui le distingue des autres forcings de la calculabilité : l'ordre partiel $(2^{<\mathbb{N}}, \succeq)$ est calculable : l'ensemble $2^{<\mathbb{N}}$ est calculable et étant donné $\sigma \in 2^{<\mathbb{N}}$, on peut calculer l'ensemble des chaînes $\tau \succeq \sigma$. Une autre de ses particularités est que la relation de forcing des contrats $\Sigma^0_1$ et $\Pi^0_1$ est respectivement $\Sigma^0_1$ et $\Pi^0_1$. Ces deux propriétés permettent d'obtenir la proposition 10-5.2 : si $\mathcal{R}$ est un contrat $\Sigma^0_n$ (resp. $\Pi^0_n$), le prédicat $\sigma \Vdash^* \mathcal{R}$ est $\Sigma^0_n$ (resp. $\Pi^0_n$). Ce contrôle très fin de la complexité de la relation de forcing permet alors un contrôle fin des sauts itérés des ensembles génériques, afin d'obtenir par exemple les résultats suivants.

1. Le corollaire 10-5.8 : si $X$ est non $\emptyset^{(n)}$-calculable et si l'ensemble $G$ est suffisamment générique, alors $G^{(n)}$ ne calcule pas $X$.
2. Le théorème 10-5.9 : si l'ensemble $G$ est suffisamment générique, on a alors $G \oplus \emptyset^{(n)} \geqslant_T G^{(n)}$.

C'est en général le genre de propriété que l'on cherche à obtenir avec n'importe quelle notion de forcing : le contrôle de la valeur de vérité de contrats arithmétiques, plus ce contrôle est fin, meilleurs sont les théorèmes que l'on peut obtenir. Malheureusement, les choses seront rarement aussi simples qu'avec le forcing de Cohen. Examinons la complexité calculatoire du forcing de Jockusch-Soare et celle du forcing de Sacks.

### 4.1. Complexité des ordres partiels

Pour parler de la calculabilité des ordres partiels d'objets abstraits, il est nécessaire de s'accorder d'abord sur leur représentation par des ensembles d'entiers. Pour le forcing de Cohen, l'ordre partiel des chaînes admet un codage bijectif naturel, alors que les notions de forcing de Jockusch-Soare et de Sacks font intervenir des objets plus complexes.

**Forcing de Jockusch-Soare.** Une idée naturelle est de confondre $\mathbb{P}$ avec l'ensemble des codes de classes $\Pi^0_1$ non vides. Notons que l'on a alors des répétitions, car plusieurs entiers codent pour la même classe, ce qui en pratique n'est pas un problème.

**Proposition 4.1.** L'ordre partiel du forcing de Jockusch-Soare est $\Pi^0_2$. ⋆

Preuve. Notons déjà que l'ensemble des codes de classes $\Pi^0_1$ non vides est $\Pi^0_1$. En effet, si une classe $\Pi^0_1$ est vide, alors l'arbre calculable $T \subseteq 2^{<\mathbb{N}}$ qui le représente n'a aucun chemin infini, et d'après le lemme de König il existe un entier $n$ tel qu'aucune chaîne de taille $n$ n'appartient à $T$, ce qui est un événement $\Sigma^0_1$.

À présent, étant donné deux classes $\Pi^0_1$ non vides $\mathcal{P}, \mathcal{Q}$, on a $\mathcal{P} \subseteq \mathcal{Q}$ ssi $\forall t\ \exists s\ \mathcal{P}[s] \subseteq \mathcal{Q}[t]$, ce qui est un prédicat $\Pi^0_2$. ∎

Notons qu'il est possible d'améliorer la complexité de l'ordre partiel du forcing de Jockusch-Soare, en ne travaillant que sur une partie bien choisie des codes de classes $\Pi^0_1$ non vides. Il existe en effet une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ telle que $f(e)$ code toujours pour une classe $\Pi^0_1$ non vide et telle que si $e$ code pour une classe $\Pi^0_1$ non vide, alors $e$ et $f(e)$ codent pour la même classe : il suffit étant donné $e$ de stopper la co-énumération de la classe $\Pi^0_1$ correspondante si celle-ci s'apprête à rendre la classe vide.

On peut également dans la pratique améliorer la complexité de l'ordre partiel en restreignant là aussi l'ensemble des codes sur lesquels on travaille : étant donné le code $e$ d'une classe $\Pi^0_1$ non vide $\mathcal{P}$, on peut considérer l'ensemble $A$ des codes de toutes les classes $\Pi^0_1$ $\mathcal{Q}$ telles que $\mathcal{P} \cap \mathcal{Q}$ est non vide. L'ensemble $A$ est $\Pi^0_1$, et contient au moins un code correspondant à chaque classe $\Pi^0_1$ non vide incluse dans $\mathcal{P}$. Cela ne permet pas bien entendu de décider si deux codes représentent des conditions de forcing comparables, mais cela simplifie la complexité calculatoire de trouver la liste de toutes les conditions de $\mathbb{P}$ se trouvant sous une condition donnée.

**Forcing de Sacks calculable.** Ici, l'idée naturelle est de confondre dans ce cadre les conditions du forcing de Sacks calculable avec les codes de fonctions $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ correspondant à des f-arbres. Là encore, le codage n'est pas injectif, mais ce n'est en pratique pas un problème.

**Proposition 4.2.** L'ordre partiel du forcing de Sacks calculable est $\Pi^0_2$. ⋆

Preuve. L'ensemble des conditions est l'ensemble des codes $e$ de fonctions $T_e : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ telles que $\forall \sigma \in 2^{<\mathbb{N}}\ \exists t \in \mathbb{N}\ T_e(\sigma)[t]\downarrow$ et telles que $\forall \sigma_0, \sigma_1 \in 2^{<\mathbb{N}}\ \sigma_0 \prec \sigma_1 \leftrightarrow T_e(\sigma_0) \prec T_e(\sigma_1)$. Ce sont bien des conditions $\Pi^0_2$ à vérifier.

Étant donné un f-arbre calculable $T$, l'ensemble des codes $e$ de f-arbres calculables $S_e$ tels que $[S_e] \subseteq [T]$ est l'ensemble des codes $e$ de f-arbre (ce qui est une condition $\Pi^0_2$) qui vérifient que pour toute chaîne $\sigma$ et pour tout entier $n$ suffisamment grand tel que $|T(\tau)| \geqslant |S_e(\sigma)|$ pour toute chaîne $\tau$ de taille $n$, il existe une chaîne $\tau$ telle que $|\tau| \leqslant n$ pour laquelle $S_e(\sigma) = T(\tau)$. Il s'agit d'une condition $\Pi^0_1$. ∎

### 4.2. Forcer des contrats $\Sigma^0_1/\Pi^0_1$

La complexité de la relation de forcing pour les contrats $\Sigma^0_1$ et $\Pi^0_1$ n'est pas directement liée à la complexité de l'ordre partiel. Nous allons voir en particulier que la relation de forcing syntaxique du forcing de Jockusch-Soare, est plus complexe que celle du forcing de Sacks.

**Forcing de Jockusch-Soare.** La complexité de la relation de forcing ne suit pas celle de la complexité des contrats à forcer, y compris déjà pour les contrats $\Sigma^0_1/\Pi^0_1$.

**Proposition 4.3.** Soit $(\mathbb{P}, \leqslant)$ le forcing de Jockusch-Soare et soit $\mathcal{R}$ un contrat. Soit $\mathcal{P}_e$ la classe $\Pi^0_1$ de code $e$, que l'on suppose non vide.

(1) Si $\mathcal{R}$ est $\Sigma^0_1$, alors le prédicat $\mathcal{P}_e \Vdash^* \mathcal{R}$ est $\Sigma^0_1$

(2) Si $\mathcal{R}$ est $\Pi^0_1$ , alors le prédicat $\mathcal{P}_e \Vdash^* \mathcal{R}$ est $\Pi^0_2$ ⋆

Preuve. Soit $T_e \subseteq 2^{<\mathbb{N}}$ un arbre calculable tel que $[T_e] = \mathcal{P}_e$.

(1) Soit $\mathcal{R}$ de la forme $\Phi(G, 0)\downarrow$ pour une fonctionnelle $\Phi$. Alors, $\mathcal{P}_e \Vdash^* \mathcal{R}$ ssi $\Phi(X, 0)\downarrow$ pour tout $X \in \mathcal{P}_e$ ssi (par la propriété de l'usage et le lemme de König) $\exists n\ \forall \sigma \in T_e \cap 2^n$ on a $\Phi(\sigma, 0)\downarrow$.

(2) Soit $\mathcal{R}$ de la forme $\Phi(G, 0)\uparrow$ pour une fonctionnelle $\Phi$. Alors, on peut obtenir $a \in \mathbb{N}$ le code de la classe $\Pi^0_1$ telle que $\mathcal{P}_a = \{X\ :\ \Phi(G, 0)\uparrow\}$. On a alors $\mathcal{P}_e \Vdash^* \mathcal{R}$ ssi $\mathcal{P}_e \subseteq \mathcal{P}_a$. Comme vue dans la preuve de la proposition 4.2, la relation d'inclusion sur les codes des classes $\Pi^0_1$ est $\Pi^0_2$. ∎

Nous verrons dans les deux sections à venir comment contourner le problème de complexité soulevé par la précédente proposition.

**Exercice 4.4. (⋆)** Soit $(\mathbb{P}, \leqslant)$ l'ordre partiel des arbres calculables $T \subseteq 2^{<\mathbb{N}}$ infinis.

Soit $\Vdash^\circ$ la relation définie par

(1) $T \Vdash^\circ \exists n \Phi(G, n)\downarrow$ s'il existe $n, t \in \mathbb{N}$ tel que pour tout $\sigma \in T$ de longueur $t$, $\Phi(\sigma, n)\downarrow$

(2) $T \Vdash^\circ \forall n \Phi(G, n)\uparrow$ si pour tout $\sigma \in T$ et tout $n < |\sigma|$, $\Phi(\sigma, n)\uparrow$

Montrer les faits suivants.

(a) $(\mathbb{P}, \leqslant)$ est un forcing de Cantor, où $[T]$ est la classe des chemins de $T$.

(b) Les ensembles suffisamment génériques pour le forcing de Jockusch-Soare et pour $(\mathbb{P}, \leqslant)$ coïncident.

(c) L'ensemble $\{T \in \mathbb{P} : T \Vdash^\circ \exists n \Phi(G, n)\downarrow$ ou $T \Vdash^\circ \forall n \Phi(G, n)\uparrow\}$ est dense dans $(\mathbb{P}, \leqslant)$.

(d) Les relations $T \Vdash^\circ \exists n \Phi(G, n)\downarrow$ et $T \Vdash^\circ \forall n \Phi(G, n)\uparrow$ sont respectivement $\Sigma^0_1$ et $\Pi^0_1$. ◇

**Forcing de Sacks calculable.** Le forcing de contrats $\Sigma^0_1$ et $\Pi^0_1$ est dans ce cas-ci de complexité minimale.

**Proposition 4.5.** Soit $(\mathbb{P}, \leqslant)$ le forcing de Sacks et soit $\mathcal{R}$ un contrat. Soit $T_e$ le f-arbre calculable de code $e$.

(1) Si $\mathcal{R}$ est $\Sigma^0_1$, alors le prédicat $T_e \Vdash^* \mathcal{R}$ est $\Sigma^0_1$.

(2) Si $\mathcal{R}$ est $\Pi^0_1$ , alors le prédicat $T_e \Vdash^* \mathcal{R}$ est $\Pi^0_1$. ⋆

Preuve. (1) Soit $\mathcal{R}$ un contrat de la forme $\Phi(G, 0)\downarrow$ pour une fonctionnelle $\Phi$. On a $T_e \Vdash^* \mathcal{R}$ ssi il existe $n, t$ tels que $\Phi(\sigma, 0)[t]\downarrow$ pour toute chaîne $\sigma \in \operatorname{Im} T_e$ de taille $n$.

(2) Soit $\mathcal{R}$ de la forme $\Phi(G, 0)\uparrow$ pour une fonctionnelle $\Phi$. On a $T_e \Vdash^* \mathcal{R}$ ssi pour tout $\sigma \in \operatorname{Im} T_2$ et pour tout $t$ on a $\Phi(\sigma, 0)[t]\uparrow$. ■

**Continuité du saut Turing.** Nous voyons à présent un théorème qui contraste avec le fait que tout suffisamment générique pour le forcing de Cohen est low généralisé. Ce n'est en général pas le cas pour les forcings à base d'arbres, et ce n'est en particulier pas le cas pour le forcing de Sacks calculable.

**Théorème 4.6**
*Soit $X$ un ensemble quelconque et soit $G$ un ensemble suffisamment générique pour le forcing de Sacks calculable. Alors, $X \oplus G \not\geq_T G'$.*

PREUVE. Soit $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ un f-arbre calculable et soit $\Phi_e$ une fonctionnelle Turing. On va construire un sous f-arbre $S$ de $T$ tel que pour chacun des chemins $G$ de $S$ on a $\Phi_e(X \oplus G, n) \neq G'(n)$ pour un certain $n$. On considère un sous f-arbre calculable $S$ de $T$ tel que pour tout $\sigma \in \operatorname{Im} S$ il existe $\tau \succeq \sigma$ avec $\tau \in \operatorname{Im} T$ et $\tau \notin \operatorname{Im} S$.

On considère à présent le code $a$ de la fonctionnelle partielle calculable telle que $\Phi_a(Y, a)\uparrow$ pour tout $Y \in [S]$ et telle que $\Phi_a(Y, a)\downarrow$ pour tout $Y \notin [S]$. Notons en particulier que $\Phi_a(Y, a)\downarrow$ pour tout $Y \in [T] \setminus [S]$. Supposons d'abord que $\Phi_e(X \oplus \sigma, a) \uparrow \notin \{0, 1\}$ pour tout $\sigma \in \operatorname{Im} S$. Alors, on peut prendre $S$ comme extension de forcing de $T$ afin de forcer la partialité de $\Phi_e$ sur l'entrée $a$. Sinon, soit $\sigma \in \operatorname{Im} S$ telle que $\Phi_e(X \oplus \sigma, a) \downarrow = i$ pour $i \in \{0, 1\}$. Si $i = 0$, alors on choisit une chaîne $\tau \succeq \sigma$ telle que $\tau \in \operatorname{Im} T$ et $\tau \notin \operatorname{Im} S$, et l'on prend comme extension de forcing un sous f-arbre de $T$ dont l'image ne contient que des extensions de $\tau$. Notons que l'on a alors $\Phi_a(Y, a)\downarrow$ pour tout chemin $Y$ de notre extension de forcing. Si $i = 1$, alors on prend comme extension de forcing le sous f-arbre de $S$ dont l'image ne contient que des extensions de $\sigma$. Notons que l'on a alors $\Phi_a(Y, a)\uparrow$ pour tout chemin $Y$ de notre extension de forcing.

Dans les deux cas, on force $\Phi_e(X \oplus G, a)$ à être différent de $G'(a)$ pour tout ensemble $G$ de notre condition de forcing. ∎

Un théorème analogue pour le forcing de Jockusch-Soare ne sera pas nécessairement vrai, par exemple à cause de l'existence de classes $\Pi^0_1$ contenant un unique élément calculable. Même en se restreignant aux classes $\Pi^0_1$ ne contenant pas de points calculables, les choses ne sont pas aussi simples et dépendront de la classe $\Pi^0_1$ avec laquelle on commence le forcing.

**Exercice 4.7.** (⋆⋆⋆) Soit $\mathcal{P}$ une classe $\Pi^0_1$ non vide. Alors, $\mathcal{P}$ est *fine* (définition due à Downey [49]) si pour toute sous-classe $\Pi^0_1$ non vide $\mathcal{Q} \subseteq \mathcal{P}$, il existe une suite finie de cylindres $[\sigma_0], \ldots, [\sigma_n]$ telle que

$$\mathcal{Q} = \mathcal{P} \cap ([\sigma_0] \cup \ \ldots \ \cup [\sigma_n]).$$

1. Montrer l'existence d'une classe $\Pi^0_1$ fine parfaite (il s'agit d'un algorithme du type méthode de priorité comme exposé dans le chapitre 13).
2. Montrer que si $\mathcal{P}$ est fine et si $X \in \mathcal{P}$, alors $X \oplus \emptyset'' \geqslant_T X'$. ◇

**Exercice 4.8.** (⋆⋆) Cet exercice anticipe sur la partie II au sein de laquelle le lemme 18-3.3 devrait être utile. On considère une variante du forcing de Jockusch-Soare avec des classes $\Pi^0_1$ ne contenant que des aléatoires au sens de Martin-Löf. Montrer que pour tout $X$ et tout ensemble $Z$ suffisamment générique pour ce forcing on a $Z \oplus X \not\geqslant_T Z'$. ◇

### 4.2.1. Forcer des contrats $\Sigma^0_2/\Pi^0_2$

Les forcing à base d'arbres sont en général adaptés, non pour forcer les contrats $\Sigma^0_1$ ou $\Pi^0_1$ (comme le montre le théorème 4.6, ils ne permettent pas d'obtenir des ensembles low généralisés), mais pour forcer les contrats $\Sigma^0_2$ ou $\Pi^0_2$, sur lesquels ils fonctionnent particulièrement bien. Nous avons vu dans la section 10-4 que dans le cas du forcing de Cohen, si l'on définit la relation de forcing par « tout filtre maximal satisfait le contrat », alors l'ensemble des conditions forçant un contrat $\Pi^0_2$ ou sa négation n'est pas dense en général. Cela nous a conduit à définir la relation de forcing pour tout filtre maximal *suffisamment générique*.

Contrairement au forcing de Cohen, les forcings à base d'arbres, comme le forcing de Jockusch-Soare ou le forcing de Sacks permettent en général de trouver des conditions de forcing dont tous les membres satisfont des formules $\Sigma^0_2$ ou $\Pi^0_2$. Le lecteur pourra constater que c'est bien ce mécanisme qui est à l'œuvre pour forcer un ensemble générique à être calculatoirement dominé. Nous introduisons pour voir cela la notion de *question de forcing*, notée $?\vdash$, qui sera développée et étudiée dans les sections suivantes pour des contrats arithmétiques arbitraires.

**Définition 4.9.** Soit $\mathcal{R}$ un contrat $\Sigma^0_2$ correspondant à $\exists n\ \Phi(G,n)\!\uparrow$ pour une fonctionnelle $\Phi$.

(1) Soit $\mathcal{P}$ une condition du forcing de Jockusch-Soare. On définit

$$\mathcal{P}\ ?\vdash \exists n\ \Phi(G,n)\!\uparrow$$

s'il existe $n$ tel que $\mathcal{P} \nVdash^* \Phi(G,n)\!\downarrow$.

(2) Soit $T$ une condition du forcing de Sacks calculable. On définit

$$T\ ?\vdash \exists n\ \Phi(G,n)\!\uparrow$$

s'il existe $n$ et $\sigma$ tel que $T\restriction_\sigma \Vdash^* \Phi(G,n)\!\uparrow$, où $T\restriction_\sigma$ est le f-arbre $S$ défini par $S(\tau) = T(\sigma\tau)$. ♢

Le premier intérêt de la relation de question de forcing que nous avons définie, est sa complexité qui est la même que celle du contrat concerné.

**Proposition 4.10.** Soit $c$ une condition du forcing de Jockusch-Soare ou du forcing de Sacks. Soit $\mathcal{R}$ un contrat $\Sigma^0_2$ correspondant à $\exists n\ \Phi(G,n)\!\uparrow$ pour une fonctionnelle $\Phi$. Le prédicat $c\ ?\vdash \exists n\ \Phi(G,n)\!\uparrow$ est $\Sigma^0_2$. ⋆

Preuve. Commençons par le forcing de Jockusch-Soare. D'après la proposition 4.3, étant donné $n$, le prédicat $\mathcal{P} \Vdash^* \Phi(G,n)\downarrow$ est $\Sigma^0_1$, et le prédicat $\mathcal{P} \nVdash^* \Phi(G,n)\!\downarrow$ est donc $\Pi^0_1$. Ainsi, le prédicat $\exists n\ \mathcal{P} \nVdash^* \Phi(G,n)\!\downarrow$ est $\Sigma^0_2$, et le prédicat $\mathcal{P}\ ?\vdash \Phi(G,n)\!\uparrow$ est donc $\Sigma^0_2$.

Passons au forcing de Sacks calculable. D'après la proposition 4.5, étant donné $n$, et un f-arbre $S$, le prédicat $S \Vdash^* \Phi(G,n)\uparrow$ est $\Pi^0_1$. Ainsi, le prédicat $\exists n\ \exists\sigma\ T\restriction_\sigma \Vdash^* \Phi(G,n)\uparrow$ est $\Sigma^0_2$, et le prédicat $T\ ?\vdash \exists n\ \Phi(G,n)\uparrow$ est donc $\Sigma^0_2$. ∎

Le second intérêt de la relation de question de forcing, est qu'elle permet de *décider* si une condition $c$ pourra être étendue en une condition $d$ pour forcer un contrat $\Sigma^0_2$, ou la négation de ce contrat. De plus, dans le cas de formules $\Sigma^0_2/\Pi^0_2$, on pourra trouver une extension $d \leqslant c$ telle que le contrat sera satisfait *pour tous les éléments* de $[d]$ (comme pour les contrats $\Sigma^0_1/\Pi^0_1$).

**Proposition 4.11.** Soit $c$ une condition du forcing de Jockusch-Soare ou du forcing de Sacks et soit $\mathcal{R}$ un contrat $\Sigma^0_2$ correspondant à $\exists n\ \Phi(G,n)\uparrow$ pour une fonctionnelle $\Phi$.

1. Si $c\ ?\vdash \exists n\ \Phi(G,n)\uparrow$, alors il existe $d \leqslant c$ tel que $\exists n\ \Phi(X,n)\uparrow$ est vrai pour tout $X \in [d]$.
2. Si $c\ ?\nvdash \exists n\ \Phi(G,n)\uparrow$, alors il existe $d \leqslant c$ tel que $\forall n\ \Phi(X,n)\downarrow$ est vrai pour tout $X \in [d]$.

Dans les deux cas, on peut trouver $d$ uniformément en $c$ et $\Phi$ à l'aide de $\emptyset''$.⋆

Preuve. Commençons par le forcing de Jockusch-Soare. Soit $\mathcal{P} \subseteq 2^{\mathbb{N}}$ une classe $\Pi^0_1$ non vide.

1. Si $\mathcal{P}\ ?\vdash \exists n\ \Phi(G,n)\uparrow$, alors il existe $n$ tel que $\mathcal{P} \nVdash^* \Phi(G,n)\downarrow$. Cela signifie que la classe $\mathcal{Q} = \{X \in \mathcal{P} : \Phi(X,n)\uparrow\}$ est une classe $\Pi^0_1$ non vide. Il s'agit alors de notre extension de forcing.
2. Si $\mathcal{P}\ ?\nvdash \exists n\ \Phi(G,n)\uparrow$, alors $\mathcal{P} \Vdash^* \Phi(G,n)\downarrow$ pour tout $n$. Dans ce cas, pour tout $X \in \mathcal{P}$, on a déjà $\forall n\ \Phi(X,n)\downarrow$.

Notons que $\emptyset''$ peut décider entre les deux cas, et donc trouver l'extension de forcing appropriée. Passons au forcing de Sacks calculable. Soit à cet effet $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ un f-arbre calculable.

1. Si $T\ ?\vdash \exists n\ \Phi(G,n)\uparrow$, alors $\exists\sigma \in 2^{<\mathbb{N}}$ et $\exists n \in \mathbb{N}$ tels que $T\restriction_\sigma \Vdash^* \Phi(G,n)\uparrow$. Dans ce cas, le f-arbre $T\restriction_\sigma$ est notre extension de forcing.
2. Si $T\ ?\nvdash \exists n\ \Phi(G,n)\uparrow$, alors pour tout $\sigma \in 2^{<\mathbb{N}}$ et pour tout $n \in \mathbb{N}$ on a $T\restriction_\sigma \nVdash^* \Phi(G,n)\uparrow$. Cela implique que pour tout $\sigma \in 2^{<\mathbb{N}}$ et pour tout $n \in \mathbb{N}$, il existe $\tau \succeq \sigma$ tel que $\Phi(T(\sigma\tau),n)\downarrow$. Alors, on procède comme dans la preuve du théorème 7-5.6 pour calculer un sous f-arbre $S$ de $T$ tel que $\forall n\ \Phi(X,n)\downarrow$ pour tout $X \in [S]$.

Notons que là encore $\emptyset''$ peut décider entre les deux cas, et donc trouver l'extension de forcing appropriée. ∎

Avant de voir comment étendre la relation de question de forcing aux contrats de complexité arbitraires, voyons comment utiliser les développements obtenus jusqu'ici pour montrer que tout ensemble suffisamment générique pour le forcing de Sacks ou le forcing de Jockusch-Soare est $\text{low}_2$ généralisé.

**Théorème 4.12**
*Si $G$ est suffisamment générique pour le forcing de Sacks ou le forcing de Jockusch-Soare, il est $\text{low}_2$ généralisé dans un sens fort : $\emptyset'' \oplus G' \geqslant_T G''$.*

Preuve. Pour tout $n$, d'après le lemme 10-5.6 le contrat $n \in G''$ est $\Sigma^0_2$. D'après la proposition 4.10 et la proposition 4.11, on peut énumérer à l'aide de $\emptyset''$ un ensemble $D_1$ de conditions $c$ telles que $n \in X''$ pour tout $X \in [c]$, et un ensemble $D_2$ de conditions $c$ telles que $n \notin X''$ pour tout $X \in [c]$, le tout tel que $D_1 \cup D_2$ soit un ensemble de conditions denses. Si $G$ est suffisamment générique, il existe une condition de $c \in D_1 \cup D_2$ telle que $G \in [c]$. La condition $c$ étant un arbre, on a besoin de $G'$ pour savoir si $G \in [c]$. Une fois la condition $c$ trouvée telle que $G \in [c]$, si $c \in D_1$ alors $n \in G''$, et si $c \in D_2$ alors $n \notin G''$. ∎

### 4.2.2. Forcer des contrats $\Sigma^0_n/\Pi^0_n$

Pour les contrats plus complexes, il n'est pas possible de trouver des conditions dont tous les éléments satisfont le contrat, et il faut revenir à la définition inductive du forcing. Notre objectif est à présent de montrer un analogue du théorème 10-5.7 de préservation de la hiérarchie arithmétique. Comme la relation $\Vdash^*$ pour les forcings de Sacks et de Jockusch-Soare est trop complexe, nous allons étendre et utiliser à la place la relation de question de forcing définie précédemment pour les contrats $\Sigma^0_2$.

**Définition 4.13.** Soit $c$ une condition du forcing de Jockusch-Soare ou de Sacks et soit $\mathcal{R}$ un contrat arithmétique. On définit la relation $?\vdash$ entre $c$ et $\mathcal{R}$ comme suit :

(1) $c ?\vdash \mathcal{R}$ pour $\mathcal{R}$ un contrat $\Sigma^0_1$ ssi $c \Vdash^* \mathcal{R}$.

(2) $c ?\vdash \exists x \mathcal{R}(x)$ où $\mathcal{R}(x)$ est un contrat $\Pi^0_1$ ssi $c ?\vdash \exists x \mathcal{R}(x)$ au sens de la définition 4.9.

(3) $c ?\vdash \exists x \mathcal{R}(x)$ où $\mathcal{R}(x)$ est un contrat $\Pi^0_k$ pour $k \geqslant 2$ ssi il existe un $d \leqslant c$ et un $n \in \mathbb{N}$ tels que $d ?\vdash \mathcal{R}(n)$.

(4) $c ?\vdash \mathcal{R}$ où $\mathcal{R}(x)$ est un contrat $\Pi^0_k$ ssi $c ?\nvdash \neg\mathcal{R}$. ♢

Nous étendons à présent la proposition 4.10 et la proposition 4.11, en commençant par montrer que la relation de question de forcing est $\Sigma^0_n$ pour les contrats $\Sigma^0_n$.

**Proposition 4.14.** Soit $c$ une condition du forcing de Jockusch-Soare ou du forcing de Sacks et soit $\mathcal{R}$ un contrat $\Sigma^0_n$ (resp. $\Pi^0_n$). La relation $c ?\vdash \mathcal{R}$ est $\Sigma^0_n$ (resp. $\Pi^0_n$). ⋆

Preuve. Le cas où $\mathcal{R}$ est un contrat $\Sigma^0_1$ a été traité dans la proposition 4.3 et la proposition 4.5. Le cas où $\mathcal{R}$ est un contrat $\Sigma^0_2$ a été traité dans la proposition 4.10.

Supposons que $\mathcal{R}$ soit un contrat $\Sigma^0_{n+1}$ égal à $\exists x \mathcal{Q}(x)$, où $\mathcal{Q}(x)$ est un contrat $\Pi^0_k$ pour $k \geqslant 2$. Alors, $c ?\vdash \exists x \mathcal{Q}(x)$ ssi il existe une condition de forcing $d$ et un entier $n$ tels que $d \leqslant c$ et tels que $d ?\vdash \mathcal{Q}(n)$. Le prédicat $d \leqslant c$ est $\Pi^0_2$ (dans le cas du forcing de Jockusch-Soare et du forcing de Sacks) et le prédicat $d ?\vdash \mathcal{Q}(n)$ est par induction $\Pi^0_k$ pour $k \geqslant 2$. Il s'ensuit que le prédicat $c ?\vdash \exists x \mathcal{Q}(x)$ est $\Sigma^0_{k+1}$.

Enfin, si $\mathcal{R}$ est un contrat $\Pi^0_n$, alors $\mathcal{P} ?\vdash \mathcal{R}$ ssi $\mathcal{P} ?\nvdash \neg\mathcal{R}$, ce qui est $\Pi^0_n$ par hypothèse d'induction. ∎

On montre à présent l'extension de la proposition 4.11 : si $c ?\vdash \mathcal{R}$, alors on va pouvoir trouver une extension $d \leqslant c$ qui va forcer $\mathcal{R}$, mais attention, il s'agit ici du forcing sémantique et non du forcing syntaxique.

**Proposition 4.15.** Soit $c$ une condition du forcing de Jockusch-Soare ou de Sacks et soit $\mathcal{R}$ un contrat arithmétique. Si $c ?\vdash \mathcal{R}$, alors il existe $d \leqslant c$ tel que $d \Vdash \mathcal{R}$. ⋆

Preuve. Si $\mathcal{R}$ est un contrat $\Sigma^0_1$ ou $\Pi^0_1$, alors le résultat est trivial par définition. Si $\mathcal{R}$ est un contrat $\Sigma^0_2$ ou $\Pi^0_2$, alors il s'agit de la proposition 4.11.

Supposons $\mathcal{R} = \exists x \mathcal{S}(x)$ où $\mathcal{S}(x)$ est un contrat $\Pi^0_k$ pour $k \geqslant 2$. Supposons que $c ?\vdash \exists x \mathcal{S}(x)$. Par définition, il existe une extension $d \leqslant c$ et un entier $n \in \mathbb{N}$ tels que $d ?\vdash \mathcal{S}(n)$. Par hypothèse de récurrence, il existe un $e \leqslant d$ tel que $e \Vdash \mathcal{S}(n)$. En particulier, $e \Vdash \exists n\ \mathcal{S}(n)$.

Supposons $\mathcal{R} = \forall x \mathcal{S}(x)$, où $\mathcal{S}(x)$ est un contrat $\Sigma^0_k$ pour $k \geqslant 2$. Supposons que $c ?\vdash \forall x \mathcal{S}(x)$. Alors, $c \not?\vdash \exists x \neg\mathcal{S}(x)$. Par définition, pour toute extension $d \leqslant c$ et tout $n \in \mathbb{N}$, $d ?\nvdash \neg\mathcal{S}(n)$. Montrons que pour tout $n$, l'ensemble $D_n = \{d : d \Vdash \mathcal{S}(n)\}$ est dense sous $c$. Soient $n \in \mathbb{N}$ et $d \leqslant c$. Par définition, comme $d ?\nvdash \neg\mathcal{S}(n)$, alors $d ?\vdash \mathcal{S}(n)$. Par hypothèse d'induction, comme $d ?\vdash \mathcal{S}(n)$, alors il existe une extension $e \leqslant d$ telle que $e \Vdash \mathcal{S}(n)$. L'ensemble $D_n$ est donc dense sous $c$. Il s'ensuit par l'exercice 2.4 que l'on a $c \Vdash \mathcal{S}(n)$ pour tout $n$, donc $c \Vdash \forall x \mathcal{S}(x)$, autrement dit $c \Vdash \mathcal{R}$. ∎

Cette fameuse question de forcing va nous permettre un contrôle fin de ce que calcule des itérations arbitraires du saut Turing. C'est l'objet de la section suivante.

### 4.3. Question de forcing

Essayons d'abstraire un peu ce qui a été fait jusqu'ici, afin d'étudier la notion de question de forcing, indépendamment du forcing sur lequel on travaille.

**Définition 4.16.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor. Une *question de forcing* est une relation $?\vdash$ entre les conditions et les contrats, telle que pour toute condition $c \in \mathbb{P}$ et tout contrat arithmétique $\mathcal{R}$

(1) Si $c ?\vdash \mathcal{R}$, alors il existe une extension $d \leqslant c$ telle que $d$ force $\mathcal{R}$

(2) $c ?\vdash \mathcal{R}$ ou $c ?\vdash \neg\mathcal{R}$. ♢

Il s'ensuit de (1) et (2) que si $c ?\nvdash \mathcal{R}$, alors il existe une extension $d \leqslant c$ telle que $d$ force $\neg\mathcal{R}$ Toute relation de forcing $\Vdash$ induit une question de forcing $?\vdash$ en définissant $c ?\vdash \mathcal{R}$ pour un contrat $\Sigma^0_n$ $\mathcal{R}$ s'il existe une extension $d \leqslant c$ telle que $d \Vdash \mathcal{R}$ et $c ?\vdash \mathcal{R}$ pour un contrat $\Pi^0_n$ si $c ?\nvdash \neg\mathcal{R}$. Si la notion de forcing est calculable, comme c'est le cas pour le forcing de Cohen, la complexité de la question de forcing pour les contrats $\Sigma^0_n$ hérite de la complexité de la relation de forcing pour les mêmes contrats.

**Exercice 4.17.** Soit $(\mathbb{P}, \leqslant)$ un forcing de Cantor, et $\Vdash$ une relation de forcing. Montrer que la relation définie par $c ?\vdash \mathcal{R}$ s'il existe une extension $d \leqslant c$ telle que $d \Vdash \mathcal{R}$ est une question de forcing. ⋄

#### 4.3.1. Préservation de la hiérarchie arithmétique

Un certain nombre de propriétés de faiblesse des ensembles suffisamment génériques pour les forcings de Cantor dépendent de l'existence d'une question de forcing avec de bonnes propriétés définitionnelles. Dans ce qui suit, nous fixons un forcing de Cantor $(\mathbb{P}, \leqslant)$ ainsi qu'une question de forcing $?\vdash$. La proposition 4.19 suivante en est le premier exemple, et généralise le théorème 10-5.7.

**Définition 4.18.** Une question de forcing $?\vdash$ *préserve la hiérarchie arithmétique* si pour toute condition $c$ et tout contrat $\mathcal{R}$ $\Sigma^0_n$, la relation $c ?\vdash \mathcal{R}$ est $\Sigma^0_n$ uniformément en $c$ et $\mathcal{R}$. ♢

**Proposition 4.19.** Soit $?\vdash$ une question de forcing qui préserve la hiérarchie arithmétique. Alors, pour tout $n \geqslant 1$, pour tout ensemble $A$ qui n'est pas $\Sigma^0_n$, et pour tout ensemble $G$ suffisamment générique, $A$ n'est pas $\Sigma^0_n(G)$. ⋆

Preuve. Pour tout $e \in \mathbb{N}$, soit

$$D_e = \left\{c \in \mathbb{P} : \left(\exists m \notin A\, c \Vdash m \in W_e^{G^{(n-1)}}\right) \text{ ou } \left(\exists m \in A\, c \Vdash m \notin W_e^{G^{(n-1)}}\right)\right\}$$

Montrons que $D_e$ est un ensemble dense dans $(\mathbb{P}, \leqslant)$. Soit $c \in \mathbb{P}$, et soit

$$U = \left\{ m \in \mathbb{N} \ : \ c ?\vdash m \in W_e^{G^{(n-1)}} \right\}$$

D'après le lemme 10-5.6, le contrat $m \in W^{G^{(n-1)}}$ est $\Sigma^0_n$ uniformément en $m$, donc comme la question de forcing préserve la hiérarchie arithmétique, l'ensemble $U$ est $\Sigma^0_n$. L'ensemble $A$ n'étant pas $\Sigma^0_n$, alors la différence symétrique $U \Delta A = (U \setminus A) \cup (A \setminus U)$ n'est pas vide.

Soit $m \in U \Delta A$.

Cas 1. On a $m \in U \setminus A$. Alors, par définition de la question de forcing, il existe une extension $d \leqslant c$ telle que $d \Vdash m \in W_e^{G^{(n-1)}}$. En particulier, $d \in D_e$.

Cas 2. On a $m \in A \setminus U$. Alors, toujours par la définition de la question de forcing, il existe une extension $d \leqslant c$ telle que $d \Vdash m \notin W_e^{G^{(n-1)}}$. Là encore, $d \in D_e$.

Dans tous les cas, il existe une extension de $d$ dans $D_e$, et l'ensemble $D_e$ est donc dense. Soit $F$ un filtre suffisamment générique pour $(\mathbb{P}, \leqslant)$. Par densité de $D_e$, on peut supposer que $F$ intersecte $D_e$ pour tout $e \in \mathbb{N}$. Soit $G = \dot{F}$. Par définition de la relation de forcing, pour tout $e \in \mathbb{N}$, soit $m \in W_e^{G^{(n-1)}}$ pour un $m \notin A$, soit $m \notin W_e^{G^{(n-1)}}$ pour un $m \in A$. On en déduit donc que $A$ n'est pas $\Sigma^0_n(G)$. ∎

**Corollaire 4.20**
*Soit $?\vdash$ une question de forcing qui préserve la hiérarchie arithmétique. Alors, pour tout $n \geqslant 0$ et tout ensemble $A$ non $\emptyset^{(n)}$-calculable, pour tout ensemble $G$ suffisamment générique, $A$ n'est pas $G^{(n)}$-calculable.*

Preuve. Comme $A$ n'est pas $\emptyset^{(n)}$-calculable, il n'est pas $\Delta^0_{n+1}$. Soit $A$ n'est pas $\Sigma^0_{n+1}$, soit $\overline{A}$ n'est pas $\Sigma^0_{n+1}$. Par la proposition 4.19, pour tout ensemble $G$ suffisamment générique pour $(\mathbb{P}, \leqslant)$, $A$ n'est pas $\Sigma^0_{n+1}(G)$ dans le premier cas, et $\overline{A}$ n'est pas $\Sigma^0_{n+1}(G)$ dans le second cas. Dans tous les cas, $A$ n'est pas $\Delta^0_{n+1}(G)$, et donc pas $G^{(n)}$-calculable. ∎

**Corollaire 4.21**
*Soit $n \geqslant 0$, et soit $A \in 2^{\mathbb{N}}$ un ensemble non $\emptyset^{(n)}$-calculable. Si $G$ est suffisamment générique pour le forcing de Jockusch-Soare ou pour le forcing de Sacks, alors $A$ n'est pas $G^{(n)}$-calculable.*

Preuve. D'après la proposition 4.14, ces deux notions de forcing préservent la hiérarchie arithmétique. ∎

### 4.3.2. Préservation d'hyperimmunité

Rappelons qu'une fonction $f : \mathbb{N} \to \mathbb{N}$ est hyperimmune relativement à $X$ si elle n'est dominée par aucune fonction $X$-calculable (voir la section 7-4). Aucune fonction n'est hyperimmune relativement à tous les degrés Turing, à commencer par le degré Turing de la fonction elle-même. Cependant, si une fonction est hyperimmune, elle est également hyperimmune relativement à tout degré calculatoirement dominé. On peut donc considérer que les degrés calculatoirement dominés « préservent » les hyperimmunités de toutes les fonctions simultanément. Nous allons maintenant étudier dans quelle mesure les ensembles suffisamment génériques pour des forcings de Cantor préservent des hyperimmunités.

Nous avons vu dans la section 10-3.1 que tout ensemble $X$ faiblement 1-générique était de degré hyperimmune. Plus précisément, la fonction principale $p_X$ de $X$ qui à $n$ associe le $n$-ième élément de $X$ est hyperimmune. Les ensembles suffisamment génériques pour le forcing de Cantor ne préservent donc pas toutes les hyperimmunités simultanément. Cependant, il est possible de s'assurer qu'ils préservent l'hyperimmunité de toute fonction hyperimmune fixée à l'avance.

**Définition 4.22.** Une question de forcing $?\vdash$ est *compacte* si pour tout $c \in \mathbb{P}$, tout contrat arithmétique $\mathcal{R}(x)$, si $c ?\vdash \exists x \mathcal{R}(x)$, alors il existe un ensemble fini $U \subseteq \mathbb{N}$ tel que $c ?\vdash \exists x \in U\ \mathcal{R}(x)$. ♢

La compacité d'une question de forcing est suffisante pour assurer la propriété de préservation d'une hyperimmunité.

**Proposition 4.23.** Soit $?\vdash$ une question de forcing compacte qui préserve la hiérarchie arithmétique. Alors, pour tout $n \geqslant 0$ et toute fonction hyperimmune $f : \mathbb{N} \to \mathbb{N}$ relativement à $\emptyset^{(n)}$, pour tout ensemble $G$ suffisamment générique pour $(\mathbb{P}, \leqslant)$, la fonction $f$ est hyperimmune relativement à $G^{(n)}$.⋆

Preuve. Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction hyperimmune relativement à $\emptyset^{(n)}$. Pour tout $e \in \mathbb{N}$, soit

$$D_e = \{c \in \mathbb{P} : (\exists m\ c \Vdash \Phi_e(G^{(n)}, m)\uparrow\ ) \text{ ou } (\exists m\ c \Vdash \Phi_e(G^{(n)}, m)\downarrow < f(m))\}.$$

Montrons alors que $D_e$ est un ensemble dense dans $(\mathbb{P}, \leqslant)$. Soit $c \in \mathbb{P}$, et soit $g : \mathbb{N} \to \mathbb{N}$, la fonction partielle qui pour tout $m$, cherche un ensemble fini $U \subseteq \mathbb{N}$ tel que $c ?\vdash \Phi_e(G^{(n)}, m)\downarrow\ \in U$. Si un tel ensemble $U$ est trouvé, $g(m) = 1 + \max U$, sinon $g(m)$ n'est pas défini. Sachant que la question de forcing préserve la hiérarchie arithmétique, $g$ est partielle $\emptyset^{(n)}$-calculable. Les deux cas suivants se présentent.

Cas 1. Il existe un $m$ tel que $g(m)$ n'est pas défini. Alors, par compacité de la question de forcing, $c ?\nvdash \Phi_e(G^{(n)}, m)\downarrow$, donc il existe un $d \leqslant c$ tel que $d \Vdash \Phi_e(G^{(n)}, m)\uparrow$. En particulier, $d \in D_e$.

Cas 2. La fonction $g$ est totale $\emptyset^{(n)}$-calculable. Par hyperimmunité de $f$ relativement à $\emptyset^{(n)}$, il existe un $m \in \mathbb{N}$ tel que $g(m) \leqslant f(m)$. En particulier, $c ?\vdash \Phi_e(G^{(n)}, m)\downarrow \in U$ pour un ensemble fini $U$ tel que $\max U < f(m)$. Par définition d'une question de forcing, il existe une extension $d \leqslant c$ telle que $d \Vdash \Phi_e(G^{(n)}, m)\downarrow \in U$, donc $\Phi_e(G^{(n)}, m)\downarrow < f(m)$. Il s'ensuit que $d \in D_e$.

Dans tous les cas, il existe une extension de $d$ dans $D_e$, donc l'ensemble $D_e$ est dense. Soit $F$ un filtre suffisamment générique pour $(\mathbb{P}, \leqslant)$. Par densité de $D_e$, on peut supposer que $F$ intersecte $D_e$ pour tout $e \in \mathbb{N}$. Soit $G = \dot{F}$. Par définition de la relation de forcing, pour tout $e \in \mathbb{N}$, soit $\Phi_e(G^{(n)})$ est une fonction partielle, soit $\Phi_e(G^{(n)}, m)\downarrow < f(m)$ pour un $m \in \mathbb{N}$, donc $f$ est hyperimmune relativement à $G^{(n)}$. ∎

La question de forcing canonique du forcing de Cohen est compacte et préserve la hiérarchie arithmétique, ce qui implique que tout ensemble suffisamment générique pour le forcing de Cohen préserve l'hyperimmunité de toute fonction préalablement fixée. On peut montrer qu'il en va de même pour les forcings de Jockusch-Soare et de Sacks.

### 4.3.3. Préservation des degrés non PA

Nous allons terminer l'étude des propriétés des questions de forcing avec un critère pour ne pas calculer de degré PA.

**Définition 4.24.** Une question de forcing $?\vdash$ est $\Pi$-*fusionnable* si pour tout $c \in \mathbb{P}$, toute paire de contrats $\Pi^0_n$ $\mathcal{R}_0, \mathcal{R}_1$ telle que $c ?\vdash \mathcal{R}_0$ et $c ?\vdash \mathcal{R}_1$, il existe une extension $d \leqslant c$ qui force simultanément $\mathcal{R}_0$ et $\mathcal{R}_1$. ♢

**Proposition 4.25.** Soit $?\vdash$ une question de forcing $\Pi$-fusionnable qui préserve la hiérarchie arithmétique. Alors, pour tout $n \geqslant 0$ pour tout ensemble $G$ suffisamment générique pour $(\mathbb{P}, \leqslant)$, $G^{(n)}$ n'est pas de degré PA relativement à $\emptyset^{(n)}$. ⋆

Preuve. D'après le théorème 8-6.2, un degré Turing est PA ssi il calcule une fonction DNC à valeurs dans $\{0, 1\}$. Dans ce qui suit, nous supposerons que $\Phi_0, \Phi_1, \ldots$ est l'énumération des fonctionnelles de Turing à valeurs dans $\{0, 1\}$. Pour tout $e \in \mathbb{N}$, soit

$$D_e = \left\{ c \in \mathbb{P} : \begin{array}{c} \\ \text{ou} \\ \\ \end{array} \begin{array}{l} (\exists m\ c \Vdash \Phi_e(G^{(n)}, m)\uparrow) \\ (\exists m\ c \Vdash \Phi_e(G^{(n)}, m)\downarrow = \Phi_e(\emptyset^{(n)}, m)) \end{array} \right\}.$$

Montrons que $D_e$ est un ensemble dense dans $(\mathbb{P}, \leqslant)$. Soit $c \in \mathbb{P}$, et soit la fonction partielle $g : \mathbb{N} \to \mathbb{N}$, qui pour tout $m$, cherche un entier $v \in \{0, 1\}$ tel que $c \mathbin{?\vdash} \Phi_e(G^{(n)}, m)\downarrow = v$. Si un tel $v$ est trouvé, $g(m) = v$, sinon $g(m)$ n'est pas défini. Sachant que la question de forcing préserve la hiérarchie arithmétique, $g$ est partielle $\emptyset^{(n)}$-calculable. Deux cas se présentent.

Cas 1. Il existe un $m$ tel que $g(m)$ n'est pas défini. Alors, pour tout $v < 2$, on a $c \mathbin{?\nvdash} \Phi_e(G^{(n)}, m) \downarrow = v$, autrement dit $c \mathbin{?\vdash} \neg(\Phi_e(G^{(n)}, m) \downarrow = v)$. Comme la relation est $\Pi$-fusionnable, il existe un $d \leqslant c$ tel que $d$ force à la fois $\neg(\Phi_e(G^{(n)}, m)\downarrow = 0)$ et $\neg(\Phi_e(G^{(n)}, m)\downarrow = 1)$ ; or, la fonctionnelle étant à valeurs dans $\{0, 1\}$, $d$ force donc $\Phi_e(G^{(n)}, m)\uparrow$. En particulier, $d \in D_e$.

Cas 2. La fonction $g$ est totale $\emptyset^{(n)}$-calculable. Sachant qu'un degré n'est pas PA relativement à lui-même, il existe $m \in \mathbb{N}$ tel que $g(m) = \Phi_m(\emptyset^{(n)}, m)$. En particulier, $c \mathbin{?\vdash} \Phi_e(G^{(n)}, m)\downarrow = g(m)$, donc par définition d'une question de forcing, il existe une extension $d \leqslant c$ telle que

$$d \Vdash \Phi_e(G^{(n)}, m)\downarrow = g(m) = \Phi_m(\emptyset^{(n)}, m).$$

Il s'ensuit que $d \in D_e$.

Dans tous les cas, il existe une extension de $d$ dans $D_e$, et l'ensemble $D_e$ est donc dense. Soit $F$ un filtre suffisamment générique pour $(\mathbb{P}, \leqslant)$. Par densité de $D_e$, on peut supposer que $F$ intersecte $D_e$ pour tout $e \in \mathbb{N}$. Soit alors $G = \dot{F}$.
Par définition de la relation de forcing, pour tout entier $e \in \mathbb{N}$, soit la fonction $m \mapsto \Phi_e(G^{(n)}, m)$ est partielle, soit $\Phi_e(G^{(n)}, m)\downarrow = \Phi_m(\emptyset^{(n)}, m)$ pour un entier $m \in \mathbb{N}$, donc $G^{(n)}$ n'est pas de degré PA relativement à $\emptyset^{(n)}$. ∎

La question de forcing pour le forcing de Cohen est $\Pi$-fusionnable, tout comme la question de forcing pour le forcing de Sacks calculable. Il n'existe en revanche pas de question de forcing $\Pi$-fusionnable pour le forcing de Jockusch-Soare, car il existe des classes $\Pi^0_1$ non vides ne contenant que des ensembles de degré PA.

**Exercice 4.26.** **(⋆)** Une question de forcing $?\vdash$ est $\Pi$-$\omega$-*fusionnable* si pour tout $c \in \mathbb{P}$, toute suite de contrats $\Pi^0_n$ $\mathcal{R}_0, \mathcal{R}_1, \ldots$ telle que $c \mathbin{?\vdash} \mathcal{R}_i$ pour tout $i \in \mathbb{N}$, il existe une extension $d \leqslant c$ qui force simultanément $\mathcal{R}_i$ pour tout $i$. Montrer que si $?\vdash$ est une question de forcing $\Pi$-$\omega$-fusionnable qui préserve la hiérarchie arithmétique, alors pour tout ensemble $G$ suffisamment générique et tout $n$, son saut itéré $G^{(n)}$ n'est pas de degré DNC relativement à $\emptyset^{(n)}$. ◇

Chapitre 12

# La quête de degrés naturels

Les débuts de la calculabilité ont permis d'observer que tous les ensembles calculatoirement énumérables provenant de problèmes naturels étaient soit calculables, soit aussi puissants que le problème de l'arrêt, qui plus est via une réduction many-one (voir la section 5-4). Cela a conduit Post à se poser en 1944 la question suivante.

**Question (Problème de Post [181]).** Existe-t-il des degrés c. e. et non calculables qui soient strictement plus faibles que le problème de l'arrêt ?⋆

Le problème de Post est resté ouvert pendant près d'une décennie, avant d'être résolue par l'affirmative par Muchnik [163] et Friedberg [63] via la méthode des priorités, que nous verrons dans la section 13-3. Le problème de Post a depuis connu bien d'autres résolutions différentes, ne faisant pas nécessairement appel à la méthode de priorité. On peut citer par exemple la construction d'un ensemble K-trivial c. e. et non calculable, que nous verrons avec le théorème 16-4.5. Toutes ces constructions reposent toutefois sur une argumentation complexe permettant de construire tout exprès un ensemble « artificiel » ayant les propriétés voulues, et les seuls problèmes de décision « naturels » indécidables connus à ce jour se réduisent au problème de l'arrêt [1]. À l'inverse, le problème de l'arrêt semble survenir naturellement un peu partout. La question s'est alors posée des propriétés qui lui confèrent cette naturalité.

1. Cette affirmation doit toutefois être reçue avec précaution, et sera atténuée dans la dernière section de ce chapitre.

# 1. Trois problèmes indécidables emblématiques

Avant d'aborder directement le problème de Post, voyons trois exemples emblématiques de problèmes de décision c. e. et indécidables, tous équivalents au problème de l'arrêt.

### Problème de correspondance de Post

Nous commençons par un problème défini par Post lui-même en 1946, que l'on appelle problème de correspondance de Post, et qui ne doit pas être confondu avec le « Problème de Post », qui fait référence à la question ci-dessus.

Étant donné deux listes finies de chaînes

$$\sigma_0, \dots, \sigma_n \in 2^{<\mathbb{N}} \quad \text{et} \quad \tau_0, \dots, \tau_n \in 2^{<\mathbb{N}},$$

existe-t-il une suite d'indices $(i_k)_{k \leqslant K}$ — possiblement avec répétition — telle que les concaténations $\sigma_{i_0}\sigma_{i_1}\dots\sigma_{i_K}$ et $\tau_{i_0}\tau_{i_1}\dots\tau_{i_K}$ forment la même chaîne ?

La question peut sembler simple en apparence, et l'on pourrait même penser au premier abord qu'il est facile de créer un algorithme permettant de la résoudre. Après tout, le nombre de chaînes en question est fini. En y réfléchissant plus longuement, le problème ne devrait pas s'avérer si évident, et pour cause, même si cela peut paraître surprenant, il s'agit d'une question indécidable, et aussi difficile que celle de savoir si un programme informatique s'arrête ou pas.

Pour le montrer, Post trouve une manière astucieuse d'encoder le calcul d'une machine de Turing via des instances du problème de correspondance. Ainsi, une instance pour laquelle une correspondance existe correspondra à un calcul qui s'arrête, et une instance pour laquelle aucune correspondance n'existe correspondra à un calcul qui se déroule à l'infini.

### Pavage du Plan

En 1961, Hao Wang imagine le problème suivant : étant donné un ensemble fini de *tuiles*, c'est-à-dire de carrés ayant une couleur sur chacun de leur côtés, l'objectif est de faire un pavage du plan en n'utilisant que des tuiles dont les couleurs correspondent à celles de l'une des tuiles de notre ensemble fini, en imposant que deux tuiles côte à côte partagent la même couleur sur leur côté en commun. Il y a bien entendu des ensembles de tuiles pour lesquels un tel pavage du plan est possible, et d'autres pour lesquels cela ne l'est pas.

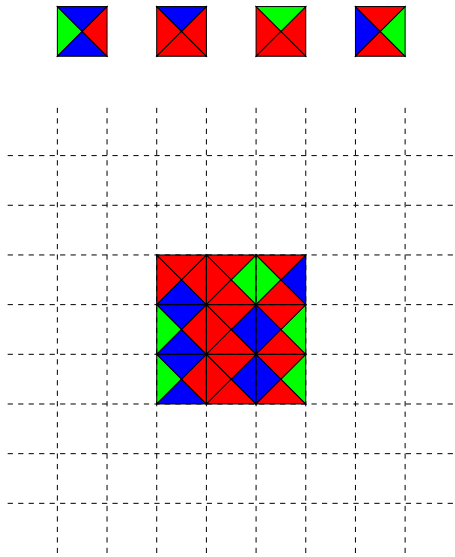


FIGURE 1.1 – *Début de pavage du plan à l'aide des quatre tuiles ci-dessus*

Un lemme découlant de celui de König s'applique aux pavages du plan utilisant un nombre fini de tuiles : si pour tout $n$ il existe un pavage de $n \times n$ tuiles, alors il existe un pavage infini du plan. On en déduit alors que si un ensemble fini de tuiles ne permet pas de paver le plan, il existe $n$ tel qu'aucun pavage de taille $n \times n$ n'est possible. Si un pavage est impossible il suffit alors de chercher le plus petit $n$ tel qu'aucune configuration de pavage de taille $n \times n$ ne fonctionne. Autrement dit, les ensembles finis de tuiles ne permettant pas de paver le plan peuvent être énumérés par un algorithme.

Wang conjecture alors qu'il en va de même pour les ensembles finis de tuiles permettant de paver le plan, ce qui rendrait décidable le problème de pavage du plan : il existerait un algorithme permettant de décider, étant donné un ensemble fini de tuiles, si ce dernier peut ou non paver le plan.

Mais en 1966, Robert Berger (un élève de Wang) montre que le problème de pavage du plan n'est pas décidable, en lui réduisant là encore le problème de l'arrêt des machines de Turing, à la manière de Post : Berger crée un système de pavage permettant de « simuler » le calcul s'effectuant sur une machine de Turing, un pavage impossible signifiant que la machine s'arrête, et un pavage possible signifiant que le calcul se poursuit indéfiniment.

**Dixième problème de Hilbert**

Une équation diophantienne est une équation polynomiale à une ou plusieurs inconnues dont les solutions sont à chercher parmi les nombres entiers — ou éventuellement rationnels —, les coefficients étant eux-mêmes également entiers. Par exemple, $a^2 + b^2 = c^2$ est une équation diophantienne ayant de nombreuses solutions comme $a = 3$, $b = 4$ et $c = 5$. Il y a en revanche des équations diophantiennes n'ayant aucune solution. Certaines d'entre elles ont demandé pour leur résolution — pour en trouver les solutions ou montrer qu'elles n'en ont pas — des efforts considérables de nombreux mathématiciens sur plusieurs siècles. L'exemple par excellence est certainement le fameux « dernier théorème de Fermat » qui stipule que pour tout entier $n > 2$, l'équation $a^n + b^n = c^n$ n'a pas de solutions entières (non triviales).

Fermat énonce son théorème en marge d'une traduction des « Arithmétiques de Diophante », dans laquelle il écrit : « *Au contraire, il est impossible de partager soit un cube en deux cubes, soit un bicarré en deux bicarrés, soit en général une puissance quelconque supérieure au carré en deux puissances de même degré : j'en ai découvert une démonstration véritablement merveilleuse que cette marge est trop étroite pour contenir.* »

De nombreux mathématiciens ont cherché cette démonstration merveilleuse des siècles durant et sans succès. C'est seulement après 357 ans d'efforts que le mathématicien Andrew Wiles, aidé de son ex-étudiant Richard Taylor apportera une preuve en utilisant des outils mathématiques évidemment bien plus complexes que ceux qui existaient à l'époque de Fermat.

En 1900, Hilbert place la question de la résolution des équations diophantiennes en dixième position dans sa fameuse liste de vingt-trois problèmes : « *On donne une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers : on demande de trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra distinguer si l'équation est résoluble en nombres entiers.* »

Hilbert demande, avant que l'on en ait une définition formelle, l'existence d'un *algorithme* permettant de savoir si toute équation diophantienne admet ou non une solution.

Là encore, l'ensemble des équations diophantiennes ayant une solution est calculatoirement énumérable ; il suffit en effet de chercher parmi tous les candidats potentiels si l'un d'eux est une solution. Martin Davis, Hilary Putnam et Julia Robinson eurent l'idée de montrer que le dixième problème de Hilbert est indécidable en suivant une intuition audacieuse, et qui s'avérera juste : une équation diophantienne est la brique de base d'une formule de l'arithmétique, en l'occurrence l'égalité entre deux termes. Aussi

Gödel a-t-il montré que les ensembles c. e. sont exactement ceux qui peuvent se définir par des formules $\Sigma_1^0$ de l'arithmétique.

Ne serait-il pas possible de transformer une telle formule en une formule $\Sigma_1^0$ équivalente $\exists x_1 \ \ldots \exists x_n \ F(x_1, \ldots, x_n)$, mais où $F$ ne soit plus qu'une grosse équation diophantienne ? Considérons par exemple le cas de la formule

$$t_1(a_1, \ldots, a_i) = q_1(b_1, \ldots, b_j) \vee t_2(x_1, \ldots, x_k) = q_2(y_1, \ldots, y_l),$$

où $t_1, t_2$ et $q_1, q_2$ sont des termes. Alors, cette formule est vraie dans $\mathbb{N}$ si, et seulement si, la formule

$$t_1(a_1, \ldots, a_i) - q_1(b_1, \ldots, b_j) = 0 \vee t_2(x_1, \ldots, x_k) - q_2(y_1, \ldots, y_l) = 0$$

est vraie dans $\mathbb{Z}$, ou ce qui revient au même si l'équation diophantienne

$$\big(t_1(a_1, \ldots, a_i) - q_1(b_1, \ldots, b_j)\big) \times \big(t_2(x_1, \ldots, x_k) - q_2(y_1, \ldots, y_l)\big) = 0$$

admet des solutions. On montre sans peine quelque chose de similaire pour le connecteur $\wedge$, la difficulté restante étant dans la suppression des quantifications existentielles et universelles bornées. Davis, Putnam et Robinson réussirent à supprimer les quantifications bornées au prix de l'utilisation de la fonction exponentielle dans les équations résultantes. Le travail sera alors achevé par Matiiassevitch, qui parvint à encoder la fonction exponentielle en équations diophantiennes, conduisant au théorème suivant.

**Théorème 1.2 (Théorème MRDP)**
*Soit $A \subseteq \mathbb{N}$ un ensemble calculatoirement énumérable. Alors, il existe une formule $\Sigma_1^0$ de l'arithmétique $F(x) = \exists y_1, \ldots, \exists y_n \ G(x, y_1, \ldots, y_n)$, où $G$ n'a pas de quantificateur tel que $x \in A$ si, et seulement si, $\mathbb{N} \vDash F(x)$.*

Le théorème MRDP est remarquable en ce sens qu'il illustre le fait que l'indécidabilité de l'arithmétique de Peano est dissimulée dans la structure même de l'addition et de la multiplication des entiers, et ce sans aucun besoin de recourir aux quantifications bornées. Il apporte bien sûr une réponse au dixième problème de Hilbert : si un algorithme permet de savoir si une équation diophantienne a des solutions entières, alors on peut créer un algorithme calculant le problème de l'arrêt.

## 2. Approche pour la naturalité des degrés Turing

Revenons à notre question d'origine : qu'y a-t-il de spécial à propos du problème de l'arrêt pour que tous les problèmes de décision c. e. lui soient équivalents ? Qu'est-ce qui fait au fond la naturalité d'un degré Turing ?

Steel [217] suggère une réponse : un degré Turing naturel devrait être définissable, et sa définition devrait se relativiser à n'importe quel degré. Par exemple, le problème de l'arrêt $\emptyset'$, initialement défini comme un ensemble particulier, à savoir $\{e : \Phi_e(e)\downarrow\}$, se relativise à tout ensemble $X$, en considérant l'ensemble $X' = \{e : \Phi_e(X, e) \downarrow\}$. Comme déjà vu dans la section 4-6, il s'agit d'une notion sur les degrés Turing, dans le sens où si $X \equiv_T Y$, alors $X' \equiv_T Y'$.

### 2.1. Question de Sacks

L'idée de Steel fait écho à une vieille question de Sacks [188] : existe-t-il une solution au problème de Post qui soit invariante sur les degrés Turing ? On dira que $W$ est un *opérateur c. e.* si $W$ correspond à une fonctionnelle Turing qui uniformément en un ensemble $X$ énumère un ensemble, que l'on notera $W^X$.
Sacks demande s'il existe un opérateur c. e. $W$ tel que $X <_T W^X <_T X'$ pour tout $X$ et tel que $X_0 \equiv_T X_1$ implique $W^{X_0} \equiv_T W^{X_1}$ pour tout $X_0, X_1$.

En y travaillant un peu, le lecteur pourra constater que la méthode de priorité utilisée dans le théorème 13-3.1 pour construire un ensemble c. e. $Y$ tel que $0 <_T Y <_T \emptyset'$ peut se relativiser à n'importe quel oracle $X$ pour obtenir un ensemble $X$-c. e. $Y$ tel que $X <_T Y <_T X'$. En revanche, il est beaucoup plus incertain que cette relativisation soit invariante dans les degrés Turing, et de fait elle ne l'est pas. Le premier résultat dans cette direction fut obtenu par Lachlan, qui apporta une réponse négative à la question de Sacks, dans le cas particulier où l'on attend de l'invariance qu'elle soit uniforme, c'est-à-dire que l'on demande l'existence de fonctions $h_1, h_2$ telles que si $\Phi_{a_1}(X_1) = X_2$ et $\Phi_{a_2}(X_2) = X_1$, alors $\Phi_{h_1(a_1)}(W^{X_1}) = W^{X_2}$ et $\Phi_{h_2(a_2)}(W^{X_2}) = W^{X_1}$. Notons que Lachlan ne demande pas que les fonctions $h_1, h_2$ soient calculables, mais simplement qu'elles existent.

Dans le cas où l'opérateur est invariant dans les degrés Turing, la notation $W(\mathbf{a})$ pour un degré Turing $\mathbf{a}$ admet un sens : il s'agit du degré Turing obtenu en appliquant à $W$ un ensemble quelconque dans le degré $\mathbf{a}$. Lachlan montre en fait la chose suivante : pour tout opérateur c. e. $W$ uniformément invariant tel que $W(\mathbf{d}) \geqslant \mathbf{d}$ pour tout degré $\mathbf{d}$, il existe un degré $\mathbf{a}$ tel que pour tout degré $\mathbf{d} \geqslant \mathbf{a}$ on a $W(\mathbf{d}) = \mathbf{d}$, ou bien tel que pour tout degré $\mathbf{d} \geqslant \mathbf{a}$ on a $W(\mathbf{d}) = \mathbf{d}'$. Dans le premier cas, on dira que $W$ coïncide avec l'identité *sur un cône*, et dans le second que $W$ coïncide avec le saut Turing *sur un cône*. L'expression *sur un cône* signifiant alors sur un cône dans les degrés Turing, c'est-à-dire sur tous les degrés supérieurs à $\mathbf{a}$ pour un certain $\mathbf{a}$, le degré $\mathbf{a}$ étant *la base du cône*. Lachlan obtient donc le résultat suivant.

**Théorème 2.1 (Lachlan [132])**
*Soit $W$ un opérateur c. e. uniformément invariant tel que $W(\mathbf{d}) \geqslant \mathbf{d}$ pour tout degré $\mathbf{d}$. Alors, $W$ est l'opérateur de saut sur un cône ou bien $W$ est l'opérateur identité sur un cône.*

### 2.2. Question de Sacks pour des degrés quelconques

Nous avons jusqu'à présent parlé des degrés c. e. naturels, en argumentant sur le fait que $\mathbf{0}$ et $\mathbf{0}'$ sont les seuls d'entre eux. Il existe toutefois beaucoup de problèmes de décision naturels strictement plus puissants que le problème de l'arrêt, et qui ne sont bien entendu pas c. e. On peut alors pousser la question de la naturalité pour des degrés quelconques. Voyons d'abord quelques exemples canoniques.

1. P : le problème de déterminer si un polynôme de $\mathbb{Z}[x, y_0, y_1, y_2, \dots]$ a des solutions pour tout élément $x$ suffisamment grand.
2. T : le problème de savoir si un énoncé de l'arithmétique est vrai dans $\mathbb{N}$.
3. WF : le problème de savoir si un arbre calculable de $\mathbb{N}^{<\mathbb{N}}$ a des chemins infinis.

Le problème P est $\Sigma^0_3$, et le triple arrêt s'y réduit de manière many-one. Le problème T permet de many-one calculer $\emptyset^{(n)}$ pour tout $n \in \mathbb{N}$ uniformément en l'entier $n$. Il est par ailleurs many-one calculable avec l'oracle $\emptyset^{(\omega)} = \oplus_n \emptyset^{(n)}$ et correspond donc au premier niveau transfini dans la hiérarchie des sauts Turing (nous verrons cela formellement dans la partie IV). Le problème WF est, quant à lui, plus complexe encore, et correspond à un saut Turing transfini plus élevé, que l'on peut noter $\emptyset^{(\omega_1^{ck})}$ et que l'on appelle communément *l'hypersaut* (nous verrons également cela formellement dans la partie IV).

Notons que les itérations du saut Turing, tout comme le saut Turing simple, se relativisent également à tout oracle de manière invariante sur les degrés Turing : par exemple, si $X \equiv_T Y$, alors $X^{(3)} \equiv_T Y^{(3)}$. Les opérateurs de saut itéré sont donc eux aussi naturels, et l'on peut trouver pour chacun d'entre eux des problèmes de décision qui leur correspondent. On peut parallèlement itérer la question de Sacks : les solutions du problème de Post ne sont pas les seules constructions de degrés exotiques en calculabilité, et cette question de l'invariance des opérateurs peut être étendue à toute construction. Considérons par exemple la construction d'un ensemble calculatoirement dominé du théorème 7-5.6 ou celle du théorème 8-4.5. Dans les deux cas, la construction nécessite $\emptyset''$. On vérifie sans peine avec l'une ou l'autre construction que l'on peut créer un opérateur $\emptyset''$-calculable $W$

tel que pour tout $X$ l'ensemble $W^X$ vérifie $X <_T W^X$ et est tel que $W^X$ est calculatoirement dominé relativement à $X$. Un tel opérateur peut-il être invariant dans les degrés Turing ? Si ce n'est pas le cas, peut-on en obtenir un calculable en $\emptyset'''$ ou en un oracle plus puissant ?

### 2.3. Conjecture de Martin

Inspiré par ces questions, et peut-être par le résultat de Lachlan sur les opérateurs c. e., Martin propose alors une conjecture pour le moins hardie qui dit essentiellement : l'opérateur de saut et ses itérations, sont les seuls opérateurs définissables et invariants dans les degrés Turing. Plus formellement, la conjecture comporte deux parties distinctes.

**Conjecture 2.2 (Martin [1] p. 281).** Soit $f : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ une fonction borélienne et invariante dans les degrés Turing. Alors, en voyant $f$ comme une fonction sur les degrés Turing, on a ce qui suit :

I. soit $f$ est constante sur un cône, soit $f$ est croissante sur un cône ;

II. si $f$ est croissante sur un cône, alors elle correspond à l'une des itérations (possiblement zéro ou transfinie) du saut Turing. ⋆

Que veut-on dire par $f$ borélienne ? Avant d'y répondre, notons que si l'on sort complètement du cadre de la calculabilité, on peut parfaitement définir des fonctions $f$ invariantes dans les degrés Turing telles que $\mathbf{a} < f(\mathbf{a}) < \mathbf{a}'$ pour tout $\mathbf{a}$, à l'aide simplement de l'axiome du choix de la théorie des ensembles. L'objectif de restreindre la conjecture aux fonctions $f$ boréliennes a essentiellement pour but d'interdire l'utilisation de cet axiome, et la conjecture est généralement présentée sans la restriction pour $f$ d'être borélienne, et avec des hypothèses supplémentaires sur les axiomes de la théorie des ensembles, signifiants essentiellement : « pour toute fonction $f$ que l'on peut définir sans utiliser l'axiome du choix ».

La conjecture de Martin nous dit en substance que les seuls opérateurs naturels et non constants dans les degrés Turing sont l'opérateur identité, l'opérateur de saut et ses itérations. Si cette conjecture est à ce jour encore ouverte, de nombreux progrès ont été faits, en y ajoutant comme le fit Lachlan la condition d'uniformité dans l'invariance des fonctions $f$ considérées. D'abord, Steel [217] a montré la partie II de la conjecture de Martin, pour les fonctions uniformément invariantes, généralisant donc considérablement le résultat de Lachlan. Ensuite, toujours pour les fonctions uniformément invariantes, Slaman et Steel [205] ont montré I, et ont aussi réussi à se passer, via une preuve très astucieuse, de l'uniformité pour le cas des fonctions $f$ telles que $f(\mathbf{a}) < \mathbf{a}$ pour tout degré $\mathbf{a}$ sur un cône : ces fonctions là sont nécessairement constantes sur un cône.

On ne sait à ce jour pas grand-chose sur la conjecture dans le cas général, et c'est par ailleurs également valable pour la question de Sacks potentiellement bien plus simple, qui elle aussi reste ouverte dans le cas de la non-uniformité. Cette histoire de non-uniformité a toutefois de quoi instiller le doute : si l'on cherche au fond à construire des fonctions invariantes dans les degrés Turing, on arrive toujours à une invariance uniforme. Cette constatation a conduit Steel à faire la conjecture suivante :

**Conjecture 2.3 (Steel [217]).** Si une fonction borélienne est invariante dans les degrés Turing, alors elle est uniformément invariante. ⋆

Si la conjecture de Steel est vraie, cela montrera alors la conjecture de Martin.

## 3. Problèmes de masse

Face au problème de Post, nous avons vu une première réponse négative, à savoir que sous des hypothèses de naturalité, il n'existe pas de degrés c. e. non calculables intermédiaires. Il existe une autre approche, complémentaire de la première, qui consiste à dire que si les itérations du saut Turing sont les seuls degrés naturels, c'est à cause de la nature trop restrictive d'un degré Turing : il existe des puissances calculatoires qui ne correspondent pas à des degrés Turing pris individuellement, mais à des classes de degrés.

Considérons par exemple les complétions de l'arithmétique de Peano. Nous avons vu avec le théorème 9-3.10 une preuve que toute théorie complète et cohérente qui étend l'arithmétique de Peano est incalculable, mais nous ne l'avons pas fait en montrant que le problème de l'arrêt pouvait se réduire à une telle théorie, et pour cause ce n'est pas toujours le cas : il existe des degrés PA $\Delta^0_2$ ne calculant pas le problème de l'arrêt. Ce n'est pas incompatible avec la conjecture de Martin, dans le sens où si l'on cherche à définir une extension naturelle bien spécifique de l'arithmétique de Peano qui soit complète et cohérente, on en trouvera une qui calcule le problème de l'arrêt ou plus. C'est le cas par exemple de l'ensemble des formules vraies dans $\mathbb{N}$.

Les degrés PA forment une puissance calculatoire *naturelle* en tant que classe : pour tout arbre binaire infini calculable, il existe une procédure qui prend une complétion de l'arithmétique de Peano en entrée, et calcule un chemin de l'arbre en retour. On sort ici de la naturalité des degrés, pour considérer à la place la naturalité des classes de degrés. Comme nous l'avons vu dans ce livre, il existe une grande variété de classes de degrés toutes définies de manière très naturelle, et qui ne correspondent pas à des itérations de l'arrêt : les degrés high, hyperimmunes, PA, ... La notion de

classe de degrés est en général abordée via le principe des *problèmes de masse*. Ces derniers remontent à Kolmogorov [119], qui en parle informellement comme d'une formalisation des principes de logique intuitionniste de Brouwer, avant d'être définis de manière rigoureuse par Medvedev [154] et Muchnik [165].

**Définition 3.1.** Un *problème de masse* $\mathcal{P} \subseteq 2^{\mathbb{N}}$ est vu comme l'ensemble de ses solutions possibles, identifiées, via un codage approprié, à des éléments de $2^{\mathbb{N}}$. ♢

Un problème sera par exemple résoluble s'il contient un élément calculable. Les problèmes sont étudiés en particulier via les rapports de force qu'ils entretiennent les uns par rapport aux autres. Muchnik propose l'approche suivante.

**Définition 3.2 (Réduction de Muchnik).** Un problème de masse $\mathcal{P}$ se réduit au sens de Muchnik à un problème de masse $\mathcal{Q}$, auquel cas on note $\mathcal{P} \leqslant_w \mathcal{Q}$, si toute solution de $\mathcal{Q}$ permet de calculer une solution de $\mathcal{P}$. ♢

Par exemple, n'importe quelle classe $\Pi^0_1$ non vide se réduit au sens de Muchnik au problème constitué des extensions complètes et cohérentes de PA. Medvedev propose lui une définition plus restrictive en demandant l'uniformité.

**Définition 3.3 (Réduction de Medvedev).** Un problème de masse $\mathcal{P}$ de réduit au sens de Medvedev à un problème de masse $\mathcal{Q}$, auquel cas on note $\mathcal{P} \leqslant_s \mathcal{Q}$, s'il existe une fonctionnelle $\Phi$ telle que $\Phi(X) \in \mathcal{P}$ pour tout $X \in \mathcal{Q}$. ♢

Toute classe $\Pi^0_1$ non vide se réduit également au sens de Medvedev à la classe des ensembles PA. Les deux notions ne coïncident cependant pas dans le cas général. Jockusch [105] a par exemple montré que la classe des fonctions $\mathrm{DNC}_2$ se réduisait au sens de Muchnik à celle des fonctions $\mathrm{DNC}_3$, mais pas au sens de Medvedev.

Les classes d'équivalence des relations $\equiv_w$ et $\equiv_s$ (dont les définitions découlent de $\leqslant_w$ et $\leqslant_s$) sont respectivement les *degrés Muchnik* et *degrés Medvedev*, dont la structure a été très étudiée. Il existe un plongement naturel des degrés Turing vers les degrés Muchnik et Medvedev, en associant à un degré Turing $\deg_T(X)$ le degré Muchnik ou Medvedev du problème $\{X\}$. Ce plongement respecte la structure de demi-treillis. On peut donc considérer les degrés Muchnik et Medvedev comme une extension des degrés Turing. En particulier, on peut définir $\mathbf{0}_w$ et $\mathbf{0}'_w$, les degrés Muchnik respectifs de $\{\emptyset\}$ et $\{\emptyset'\}$. La proposition suivante est donc d'une certaine manière liée à la question originale de Post.

**Proposition 3.4.** Soit PA le degré Muchnik des degrés PA. Alors,

$$\mathbf{0}_w <_w \mathrm{PA} <_w \mathbf{0}'_w. \qquad \star$$

Cette généralisation des degrés Turing a un coût : les degrés Muchnik et Medvedev sont bien plus nombreux. Plus précisément, les degrés Turing ont la puissance du continu ($|2^{\mathbb{N}}|$), tandis que les degrés Muchnik et Medvedev ont pour cardinalité $\left|2^{2^{\mathbb{N}}}\right|$ et possèdent des anti-chaînes de cette taille.

Chapitre 13

# Méthode de priorité et degrés c. e.

Parmi les ensembles non calculables, les ensembles calculatoirement énumérables jouent un rôle particulièrement important. Ces ensembles sont « presque calculables » au sens où si un entier $n$ appartient à un ensemble c. e. $A$, alors cette information sera connue en un temps fini. La classe des ensembles calculatoirement énumérables est assez naturelle, pour plusieurs raisons.

Tout d'abord, les ensembles c. e. possèdent plusieurs caractérisations très différentes, ce qui en fait une classe relativement *robuste*. Par définition, un ensemble est c. e. s'il est le domaine d'une fonction partielle calculable. Les ensembles c. e. non vides sont également précisément ceux qui sont l'image d'une fonction totale calculable, la fonction pouvant être injective si l'ensemble est infini (voir la proposition 3-7.2). De manière équivalente, un ensemble est c. e. si, et seulement si, il est réductible au problème de l'arrêt par une réduction many-one, ou encore s'il est $\Sigma^0_1$.

Les ensembles calculatoirement énumérables forment une *classe syntaxique* contrairement aux ensembles calculables. En effet, les ensembles c. e. sont précisément les ensembles $\Sigma^0_1$, tandis que les ensembles calculables sont les ensembles définissables à la fois par un prédicat $\Sigma^0_1$ et $\Pi^0_1$. Cette nature syntaxique donne aux ensembles c. e. de meilleures propriétés d'uniformité. Ainsi, la classe des ensembles calculatoirement énumérables est uniformément c. e., car il suffit de lister toutes les fonctions partielles calculables, ou de manière équivalente toutes les formules $\Sigma^0_1$. Les ensembles calculables ne peuvent en revanche pas être listés de manière calculable (d'après le théorème 7-6.2, les degrés high sont exactement ceux permettant de lister les ensembles calculables).

Enfin, et c'est peut-être un des arguments les plus importants en faveur de la naturalité des ensembles c. e., un certain nombre de problèmes non décidables en mathématiques se trouvent être calculatoirement énumérables. Parmi eux, on citera bien sûr le problème de l'arrêt, mais également l'ensemble des théorèmes de l'arithmétique (voir le théorème 9-3.7), ou même l'ensemble des solutions d'équations diophantiennes (voir, à cet effet, le théorème 12-1.2).

## 1. Degrés c. e.

Être calculatoirement énumérable est une propriété d'ensemble et non de degré Turing. En effet, nous avons vu que les degrés Turing sont clos par complémentaire, or par la proposition 3-7.4, si un ensemble et son complémentaire sont c. e., alors ils sont calculables. En particulier, le problème de l'arrêt $\emptyset'$ est c. e., mais est Turing-équivalent à son complémentaire, qui lui ne l'est pas. Nous avons cependant défini une notion de degré c. e. au début de la section 7-3, définition que nous répétons ci-après.

**Définition 1.1.** Un degré Turing est *c. e.* s'il contient un ensemble calculatoirement énumérable. ♢

Nous avons vu que les degrés Turing n'étaient pas bornés, car pour tout degré $\mathbf{d}$, son saut Turing $\mathbf{d}'$ est strictement au-dessus. Les degrés c. e., en revanche, sont bornés par $\mathbf{0}'$.

**Proposition 1.2.** Les degrés c. e. ont pour degré maximum $\mathbf{0}'$. ⋆

Preuve. Par le théorème de Post (voir la proposition 5-4.3), un ensemble est c. e. si, et seulement si, il est many-one réductible à $\emptyset'$. Les réductions many-one étant des cas particuliers de réductions Turing, tout degré c. e. est Turing-réductible à $\emptyset'$. ∎

Les degrés c. e. formant un sous-ensemble des degrés Turing, aussi les questions concernant les degrés Turing se formulent-elles pour les degrés calculatoirement énumérables. Sont-ils linéairement ordonnés ? Forment-ils un ordre bien fondé ? Et, avant tout cela, existe-t-il des degrés c. e. autres que $\mathbf{0}$, le degré Turing des ensembles calculables, et $\mathbf{0}'$, le degré Turing du problème de l'arrêt ?

Les degrés c. e. sont notoirement difficiles à manipuler, en raison de la contrainte de calculabilité de leur énumération. La méthode des extensions finies n'est plus adaptée, et il faudra faire appel à des techniques très élaborées pour prouver des résultats similaires à ceux obtenus dans les degrés Turing.

## 2. Méthode de permission

La méthode de permission permet de calculer un ensemble c. e. $A$ à partir d'un autre ensemble c. e. $B$. Elle se base sur la notion de *fonction de calcul* vue à la section 4-7. Rappelons qu'étant donné une approximation c. e. $(A_s)_{s\in\mathbb{N}}$ d'un ensemble c. e. $A$ (ou plus généralement pour toute approximation $\Delta^0_2$), la fonction de calcul associée à cette approximation est la fonction $c_A : \mathbb{N} \to \mathbb{N}$ qui à $n$ associe le plus petit temps $s$ tel que $A_s \upharpoonright_n = A \upharpoonright_n$. En particulier, cette fonction est calculable en $A$. De plus, par la proposition 4-7.9, toute fonction dominant $c_A$ recalcule $A$.

**Proposition 2.1 (Méthode de permission).** Soient $A$ et $B$ des ensembles c. e. d'approximations c. e. respectives $(A_s)_{s\in\mathbb{N}}$ et $(B_s)_{s\in\mathbb{N}}$. S'il existe une fonction $B$-calculable $f : \mathbb{N} \to \mathbb{N}$ telle que pour tout $n, s \in \mathbb{N}$

$$A_{s+1} \upharpoonright_n \neq \mathcal{A}_s \upharpoonright_n \quad \Rightarrow \quad B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)},$$

alors $A \leqslant_T B$. ⋆

Preuve. Pour tout $n$, $c_A(n) \leqslant c_B(f(n))$, or $c_B \oplus f \leqslant_T B$, donc $B$ calcule une fonction dominant $c_A$. Par la proposition 4-7.9, $B$ calcule $A$. ∎

La plupart du temps, la fonction $f$ sera calculable et croissante, voire la fonction identité. Informellement, l'approximation de $A$ n'est autorisée à ajouter un élément à $A$ que si au même moment, $B$ ajoute un élément plus petit que $f(x)$. Autrement dit, l'ensemble $A$ attend la permission de $B$ pour ajouter des éléments, ce qui donne son nom à cette méthode. La méthode de permission se combine souvent avec d'autres techniques comme la méthode de priorité que nous verrons dans la section suivante. La méthode de permission ne perd pas en généralité, au sens où l'on peut prouver la réciproque suivante dans le cas où l'ensemble $B$ est infini.

**Proposition 2.2.** Soient $A$ et $B$ des ensembles c. e. tels que $A \leqslant_T B$ *et* $B$ *est infini*. Alors, il existe des approximations c. e. $(A_s)_{s\in\mathbb{N}}$ et $(B_s)_{s\in\mathbb{N}}$ et une fonction $B$-calculable $f : \mathbb{N} \to \mathbb{N}$ telle que pour tout $n, s \in \mathbb{N}$

$$A_{s+1} \upharpoonright_n \neq \mathcal{A}_s \upharpoonright_n \quad \Rightarrow \quad B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)} . \qquad ⋆$$

Preuve. Soient $(A_s)_{s\in\mathbb{N}}$ et $(B_s)_{s\in\mathbb{N}}$ des approximations c. e. de respectivement $A$ et $B$. Sachant que $B$ est infini, en accélérant son approximation c. e., on peut supposer sans perte de généralité que $B_{s+1} \neq B_s$ pour tout $s$. Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction qui à $n$ associe le plus petit entier $m$ tel que $B_{s+1} \upharpoonright_m \neq B_s \upharpoonright_m$ pour tout $s \leqslant c_A(n)$. La fonction $f$ est $c_A \oplus B$-calculable, or $c_A \leqslant_T A \leqslant_T B$, donc $f \leqslant_T B$. Pour tout $n, s \in \mathbb{N}$, si $A_{s+1} \upharpoonright_n \neq \mathcal{A}_s \upharpoonright_n$, alors $c_A(n) \geqslant s + 1$, donc $B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)}$. ∎

## 3. Méthode de priorité $\Sigma^0_1$ (à blessure finie)

Post posa la question en 1944 [181] de savoir s'il existait des ensembles calculatoirement énumérables qui sont à la fois non calculables et Turing incomplets, c'est-à-dire qui ne permettent pas en tant qu'oracle de calculer le problème de l'arrêt. La question est restée ouverte pendant plus d'une décennie, avant d'être résolue par l'affirmative indépendamment par Muchnik [163] et Friedberg [63], qui ont introduit la fameuse *méthode de priorité*. Cette technique trouvera par la suite de très nombreuses applications pour l'étude des ensembles c. e. et $\Delta^0_2$, qui s'avèrent avoir une structure très riche, comme en témoigne le théorème de densité de Sacks : soient $X, Y$ des ensembles c. e. tels que $X <_T Y$ ; il existe alors un ensemble c. e. $Z$ tel que $X <_T Z <_T Y$.

De manière générale, la méthode de priorité sert le même but que la méthode des extensions finies (voir la section 4-8), c'est-à-dire construire des ensembles satisfaisant des propriétés de force et de faiblesse, mais cette fois-ci en contrôlant la complexité de ces ensembles dans la hiérarchie arithmétique. Cette contrainte supplémentaire est à l'origine d'une explosion de la complexité des constructions. En effet, comme dans la méthode des extensions finies, il s'agit de satisfaire une infinité de contrats simultanément, mais tandis que pour la méthode des extensions finies, la construction est omnisciente, l'argument de méthode de priorité doit s'effectuer avec une puissance calculatoire limitée. Il est donc nécessaire de poursuivre la construction de l'ensemble sans avoir une pleine connaissance de la situation. La construction se fait donc par essais-erreurs, avec des rétropédalages lorsque l'on s'aperçoit d'une erreur. Les stratégies pour satisfaire les contrats entrent en conflit, et toute la difficulté de la construction consiste à s'assurer que ces conflits et rétropédalages n'empêchent pas l'objectif global : construire un ensemble satisfaisant tous les contrats.

Nous commençons ici simplement par la plus facile des utilisations de la méthode de priorité, qui fut la première à être introduite, permettant de résoudre le problème de Post. Il s'agit d'une méthode de priorité *à blessure finie*, c'est-à-dire que chaque stratégie pour satisfaire un contrat ne devra rétropédaler qu'un nombre fini de fois avant de pouvoir réaliser son objectif.

**Théorème 3.1 (Friedberg (1957), Muchnik (1956))**
*Il existe des ensembles c. e. incomparables pour la réduction Turing.*

L'énoncé du théorème de Friedberg et Muchnick est similaire au théorème de Kleene et Post (voir la proposition 4-8.1), mais impose la contrainte supplémentaire aux ensembles d'être calculatoirement énumérables.

Preuve. Comme pour le théorème de Kleene et Post (voir la proposition 4 -8.1), nous allons construire deux ensembles, $A$ et $B$, satisfaisant chacun une propriété de force et une propriété de faiblesse. Ces propriétés se déclinent chacune sous la forme de deux suites de contrats : $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{R}_e : W_e^A \neq B \qquad \mathcal{S}_e : W_e^B \neq A.$$

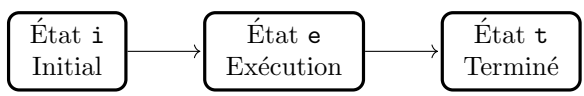
Rappelons le sens de la notation $W_e^A$, qui désigne l'ensemble c. e. relativement à $A$ de code $e$ (c'est-à-dire l'ensemble $\{n \in \mathbb{N} \ : \ \Phi_e(A, n) \downarrow\}$). Les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ assurent que $A \not\geqslant_T B$, car le complémentaire de $B$ n'est pas c. e. en $A$. Symétriquement, si les contrats $(\mathcal{S}_e)_{e\in\mathbb{N}}$ sont simultanément satisfaits, alors $B \not\geqslant_T A$.

**Remarque**

Les ensembles $A$ et $B$ étant c. e., il est équivalent de dire qu'ils ne sont pas calculables, ou que leur complémentaire n'est pas c. e. Ainsi, les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ sont sans perte de généralité. Cette formulation des contrats simplifie les notations.

Les ensembles $A$ et $B$ devant être c. e., nous allons énumérer des éléments dans $A$ et dans $B$ au cours d'un processus calculable. Plus formellement, nous allons définir deux suites uniformément calculables d'ensembles finis $A_0 \subseteq A_1 \subseteq \ldots$ et $B_0 \subseteq B_1 \subseteq \ldots$ en commençant par $A_0 = B_0 = \emptyset$. Pour des raisons de présentation, nous omettrons l'indice $s$ et parlerons de $A$ et $B$ comme des ensembles en cours de construction et évoluant au cours du temps. Nous utiliserons les indices lorsqu'il sera nécessaire de distinguer les ensembles à différentes étapes de temps. Pour chaque contrat $\mathcal{R}_e$ ou $\mathcal{S}_e$, nous allons décrire un processus responsable de sa satisfaction. Les différents processus s'exécuteront en parallèle. Un processus responsable de la satisfaction d'un contrat $\mathcal{R}_e$ (resp. $\mathcal{S}_e$) est appelé *stratégie pour* $\mathcal{R}_e$ (resp. $\mathcal{S}_e$). Dans cette construction, une seule stratégie sera nécessaire par contrat. Nous verrons plus tard des arguments de priorité qui associeront plusieurs stratégies à chaque contrat.

**Satisfaction d'un contrat $\mathcal{R}_e$.** Voici une stratégie pour construire deux ensembles c. e. $A$ et $B$ satisfaisant un unique contrat $\mathcal{R}_e$. Pour gagner en généralité et préparer la satisfaction de plusieurs contrats, nous allons également supposer que d'autres processus ou stratégies tournent en parallèle, et ajoutent des éléments à $A$ et $B$ au cours du temps. À chaque instant $t$, la stratégie pour $\mathcal{R}_e$ se retrouve dans l'un des trois états suivants :



*État* `i`. C'est l'état initial du processus. Pour sortir de cet état, le processus passe par une phase d'initialisation qui consiste à choisir un entier $x_{\mathcal{R}_e} \notin B$ unique. Ce nombre existe, car, à tout instant, les ensembles $A$ et $B$ ont un nombre fini d'éléments. Une fois $x_{\mathcal{R}_e}$ choisi, notre processus pose une *restreinte* sur $x_{\mathcal{R}_e}$, c'est-à-dire qu'il interdit aux autres processus tournant en parallèle d'ajouter $x_{\mathcal{R}_e}$ dans $B$. Seul notre processus est décisionnaire de l'ajout de $x_{\mathcal{R}_e}$. Notons qu'une fois que $x_{\mathcal{R}_e}$ est ajouté à $B$, il ne pourra plus en ressortir, car l'ensemble $B$ est c. e. Une fois la restreinte posée, le processus entre dans l'état `e`, dit d'exécution.

*État* `e`. Pendant cette phase, le processus exécute $\Phi_e^A(x_{\mathcal{R}_e})$ jusqu'à ce que l'exécution se termine. Si celle-ci ne se termine pas, le processus reste dans cet état. Notons qu'au cours de cette phase, l'ensemble $A$ peut évoluer, car d'autres processus peuvent ajouter des éléments à $A$ en parallèle. Le processus doit prendre en compte cette évolution, et calcule donc $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t]$ à l'instant $t$. Si $\Phi_e^{A_t}(x_{\mathcal{R}_e}) \downarrow$ à un instant $t$, alors le processus pose une *restreinte* sur l'usage de ce calcul, c'est-à-dire sur les bits de l'oracle $A_t$ ayant servi à cette exécution, empêchant ainsi les autres processus d'y effectuer une modification. On s'assure donc que $\Phi_e^A(x_{\mathcal{R}_e}) \downarrow$, autrement dit que $x_{\mathcal{R}_e} \in W_e^A$. Le processus ajoute alors $x_{\mathcal{R}_e}$ à $B$, de telle sorte que $W_e^A \neq \overline{B}$, et se retrouve dans l'état `t`.

*État* `t`. Dans cet état, le processus a terminé son exécution. Il ne sort pas de cet état.

**Issues**. Étudions les différentes issues de la stratégie pour $\mathcal{R}_e$. Le processus passe toujours de l'état `i` à l'état `e`, mais peut ne jamais arriver dans l'état `t`. Nous avons donc deux issues possibles. Issue `p` : il reste bloqué dans l'état d'exécution (état `e`). Dans ce cas, $\Phi_e^A(x_{\mathcal{R}_e}) \uparrow$ et $x_{\mathcal{R}_e} \notin B$, donc $W_e^A \neq \overline{B}$. On dit que le contrat $\mathcal{R}_e$ est *satisfait passivement*. Issue `a` : il entre dans l'état `t` de terminaison. Alors, par ses actions de restreintes et l'ajout de $x_{\mathcal{R}_e}$ dans $B$, il s'assure que le contrat $\mathcal{R}_e$ est satisfait par la seconde clause de la disjonction. On dit alors que le contrat $\mathcal{R}_e$ est *satisfait activement*. Dans les deux cas, le contrat $\mathcal{R}_e$ est satisfait.

**État versus issue d'une stratégie**

Il convient de bien distinguer l'état d'une stratégie, et son issue. L'état d'une stratégie dépend de chaque étape, et est une information connue à cette étape. L'issue de la stratégie est un comportement limite qui n'est pas connu en temps fini. Nous n'avons vu que des issues de la forme « La stratégie restera dans tel état au bout d'un moment », mais nous verrons d'autres issues dans la méthode de priorité à blessure infinie, comme « La stratégie passera successivement par tous ses états. »

**Conflits**. Des complications se posent lorsque l'on veut satisfaire plusieurs contrats simultanément. En effet, la stratégie d'un contrat pose des restreintes sur des bouts finis de $A$ et $B$ au cours du temps, ce qui peut entrer en conflit avec les besoins des autres stratégies. Nous allons donc analyser dans quelle mesure les stratégies peuvent entrer en conflit, que cela soit entre stratégies pour contrats de même type (par exemple $\mathcal{R}_a$ et $\mathcal{R}_b$), ou entre stratégies pour contrats de type différent (par exemple $\mathcal{R}_a$ et $\mathcal{S}_b$).

**Satisfaction de tous les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$.** De manière générale, dans les arguments de priorité, les stratégies pour les contrats de même nature sont relativement faciles à satisfaire simultanément. Les seuls conflits possibles entre deux contrats $\mathcal{R}_e$ et $\mathcal{R}_d$ surviendraient si les deux stratégies avaient choisi le même entier $x$ (autrement dit, $x_{\mathcal{R}_e} = x_{\mathcal{R}_d}$) lors de leur passage de l'état `i` à l'état `e`, et l'un des deux, disons $\mathcal{R}_e$, voit son calcul $\Phi_e^A(x)$ se terminer et cherche à ajouter $x$ dans $B$ pour passer à l'état `t` de terminaison, tandis que $\mathcal{R}_d$ est encore dans l'état `e` d'exécution. Pour éviter ces conflits, il suffit d'associer dans l'état `i` un entier $x$ différent pour chaque contrat $\mathcal{R}_e$. Cela est possible, car, $B$ étant fini au cours de la construction, il existe une infinité d'entiers hors de $B$.

**Satisfaction d'un contrat $\mathcal{R}_e$ et $\mathcal{S}_d$.** Supposons maintenant que l'on veuille satisfaire deux contrats de nature opposée. Par défaut, le contrat $\mathcal{S}_d$ jouant un rôle symétrique, la stratégie pour le satisfaire s'obtient à partir de celle pour $\mathcal{R}_e$ en substituant $A$ à $B$ et inversement.

- ▷ État `i`. Choisir un entier $x_{\mathcal{S}_d} \notin A$, et poser une restreinte sur $x_{\mathcal{S}_d}$.
- ▷ État `e`. Exécuter $\Phi_d^B(x_{\mathcal{S}_d})$. Si l'exécution s'arrête à l'instant $t$, restreindre l'usage de $\Phi_d^{B_t}(x_{\mathcal{S}_d})$ et ajouter $x_{\mathcal{S}_d}$ à $A$, puis passer dans l'état `t`.
- ▷ État `t`. Le processus est terminé.

Un nouveau conflit peut se poser entre la stratégie pour $\mathcal{R}_e$ et celle pour $\mathcal{S}_d$. Dans l'état `e`, la stratégie pour $\mathcal{R}_e$ peut voir l'exécution de $\Phi_e^A(x)$ s'arrêter avec un usage de $s$ bits d'oracle, pour $s > x_{\mathcal{S}_d}$. Il impose alors une restreinte sur $A_t\restriction_s$, interdisant aux autres processus de modifier ces valeurs. Si, plus tard, la stratégie pour $\mathcal{S}_d$ voit son exécution de $\Phi_d^B(x_{\mathcal{S}_d})$ se terminer, elle ne pourra pas ajouter $x_{\mathcal{S}_d}$ à $A$ à cause de la restreinte précédente. La stratégie pour $\mathcal{S}_d$ est *blessée* et va devoir revenir dans son état `i`, choisir un nouvel entier $x_{\mathcal{S}_d}$ suffisamment grand pour ne pas être restreint, et recommencer la procédure. La stratégie pour $\mathcal{R}_e$ étant à l'état `t` de terminaison, elle n'agira plus, et ne posera donc plus de nouvelle restreinte risquant de blesser la stratégie pour $\mathcal{S}_d$.

De manière générale, on peut satisfaire un nombre fini de contrats $\mathcal{R}_e$ et $\mathcal{S}_d$ simultanément avec la même méthode. Dès lors qu'une stratégie pose

une restreinte sur l'usage de son calcul lorsque celle-ci passe à l'état `t` et se termine, elle blesse toutes les stratégies qui n'ont pas encore atteint l'état `t`, et les fait revenir dans l'état `i`. Les stratégies ainsi blessées vont alors choisir de nouveaux éléments en dehors de toute restreinte. Notons que lorsqu'une stratégie entre dans l'état `t`, elle n'en sort plus. Les blessures étant causées uniquement par l'entrée d'une stratégie arrivant dans l'état `t`, chaque stratégie n'est blessée qu'un nombre fini de fois, et finira par entrer dans l'état `t`, ou restera bloquée dans l'état `e`.

**Satisfaction de tous les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$.** Un nouveau problème se pose lorsque l'on veut satisfaire une infinité de contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$. Supposons que la stratégie pour $\mathcal{R}_e$ choisisse dans son état `i` un entier $x_{\mathcal{R}_e}$ et se lance dans l'exécution de $\Phi_e^A(x_{\mathcal{R}_e})$, mais n'a pas le temps d'atteindre la fin de l'exécution, car la stratégie pour un contrat $\mathcal{S}_{d_0}$ voit sa propre exécution se terminer avant elle et pose une restreinte sur $x_{\mathcal{R}_e}$ pour préserver l'usage de son calcul. La stratégie pour $\mathcal{R}_e$ est alors blessée, et se retrouve à nouveau dans l'état `i` et choisit un nouvel entier $x_{\mathcal{R}_e}$ et recommence l'exécution de $\Phi_e^A(x_{\mathcal{R}_e})$ pour son nouvel $x_{\mathcal{R}_e}$. Avant d'atteindre la fin de son calcul, par manque de chance, une autre stratégie $\mathcal{S}_{d_1}$ atteint l'état `t`, et blesse encore la stratégie pour $\mathcal{R}_e$, et ainsi de suite à l'infini. La stratégie pour $\mathcal{R}_e$ changera alors infiniment souvent d'entier $x_{\mathcal{R}_e}$, et le contrat $\mathcal{R}_e$ ne sera jamais satisfait.

Pour résoudre ce problème, nous allons ordonner les stratégies. Soient $R_e$ la stratégie pour $\mathcal{R}_e$ et $S_e$ la stratégie pour $\mathcal{S}_e$. On ordonne les stratégies comme suit :

$$R_0, S_0, R_1, S_1, R_2, S_2, \dots$$

en considérant qu'une stratégie est moins prioritaire que les stratégies à sa gauche, mais plus prioritaire que celles à sa droite. Par exemple, la stratégie pour $\mathcal{S}_1$ est moins prioritaire que les stratégies pour $\mathcal{R}_0$ et $\mathcal{S}_0$, mais plus prioritaire que les stratégies pour $\mathcal{R}_2, \mathcal{S}_2$, et ainsi de suite. De la sorte, chaque stratégie est « en dessous » d'un nombre fini de stratégies, et « au-dessus » du reste.

**Remarque**

Dans la preuve du théorème de Friedberg et Muchnik, chaque contrat possède exactement une stratégie, ce qui rend le distinguo entre stratégie $R_e$ et contrat $\mathcal{R}_e$ inutile. L'ordre donné sur les stratégies induit ainsi un ordre sur les contrats. Cependant, dans les constructions suivantes, lorsque les contrats se verront attribuer plusieurs stratégies, il sera essentiel de donner un ordre total de priorité sur les stratégies et non sur les contrats.

La règle d'or que l'on établit est alors la suivante.

> Les restreintes posées par une stratégie ne s'appliquent qu'aux stratégies de plus faible priorité. Ainsi, une stratégie ne peut être blessée que par les stratégies de plus forte priorité, et peut blesser toutes les stratégies de plus faible priorité.

En supposant qu'une stratégie ne pose qu'un nombre fini de restreintes avant d'atteindre son état terminal, et cesse d'agir au bout d'un moment si elle n'est pas blessée, une simple induction montre que chaque stratégie n'est blessée qu'un nombre fini de fois. Contrairement à la satisfaction d'un nombre fini de contrats simultanément, il se peut qu'un processus arrive à l'état `t`, mais soit ensuite blessé par une stratégie de plus forte priorité qui aura ignoré la restreinte posée. Heureusement, au bout d'un moment, la stratégie de plus forte priorité cessera de blesser la plus faible, qui finira par se stabiliser.

**Construction**. Formellement, la construction se fait par étapes $t$=0,1,…, et chaque étape est divisée en sous-étapes $s < t$. Initialement, toutes les stratégies sont dans l'état `i`. À l'étape $t \geqslant 0$ et la sous-étape $s < t$, on considère le contrat $\mathcal{R}_e$ si $s = 2e$ et $\mathcal{S}_e$ si $s = 2e + 1$.

Au début de la sous-étape $s = 2e$, la stratégie pour $\mathcal{R}_e$ a les trois états possibles suivants.

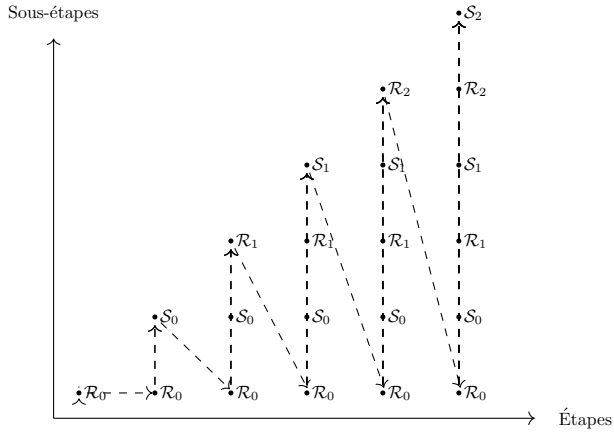
(`i`) La stratégie choisit un entier $x_{\mathcal{R}_e} \notin B_t$ qui n'a pas été restreint par une stratégie de plus grande priorité, et pose une restreinte dessus. Elle se retrouve alors dans l'état `e`.

(`e`) La stratégie exécute $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t]$. Si $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t] \downarrow$, alors elle pose une restreinte sur tous les bits ayant servi au calcul de $\Phi_s^{A_t}(x_{\mathcal{R}_e})[t]$, et blesse toutes les stratégies de plus faible priorité en les faisant retourner à l'état `i`. Elle se retrouve alors dans l'état `t`. Si $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t] \uparrow$, la stratégie reste dans l'état `e`.

(`t`) La stratégie n'agit pas et reste dans cet état.

À la sous-étape $s = 2e+1$, on applique la même procédure pour $\mathcal{S}_e$ *mutatis mutandis*, puis on passe à la sous-étape suivante, jusqu'à atteindre $t$, auquel cas on passe à l'étape $t + 1$, et ainsi de suite.

Cela conclut la construction.




**Vérification.** Tout d'abord, remarquons que la construction décrite plus haut est bien un processus calculable qui énumère deux ensembles $A$ et $B$. En particulier, une fois un élément énuméré dans $A$ ou $B$, on ne change pas d'avis et il y reste. Remarquons aussi que si une stratégie pour un contrat n'est plus blessée après une étape $r$, alors le contrat associé sera satisfait passivement ou activement à la fin de la construction. La difficulté réside dans le fait de montrer que pour tout contrat il existe bien un tel $r$.

Montrons par induction sur $e \in \mathbb{N}$ que les stratégies pour les contrats $\mathcal{R}_e$ et $\mathcal{S}_e$ blessent finiment souvent des stratégies de priorité inférieure. Supposons par hypothèse d'induction qu'il existe une étape $t$ après laquelle aucune des stratégies pour les contrats $\mathcal{R}_d$ et $\mathcal{S}_d$, avec $d < e$, ne blessent des stratégies de priorité inférieure. En particulier, à partir de l'étape $t$, la stratégie pour $\mathcal{R}_e$ ne sera plus blessée, et soit reste désormais dans l'état d'exécution (état `e`), soit passe au bout d'un moment dans l'état de terminaison (état `t`) en ne blessant plus qu'une dernière fois les stratégies de priorité inférieure. Le même raisonnement s'applique à la stratégie pour le contrat $\mathcal{S}_e$. Ainsi, chaque stratégie blesse finiment souvent les stratégies de priorité inférieure, et chaque contrat finira par être satisfait, passivement ou activement. Cela conclut la preuve du théorème 3.1. ∎

**Remarque**

La preuve précédente était très détaillée afin de donner les intuitions de la méthode de priorité. Nous irons désormais plus directement vers la construction finale dans les autres preuves. Il est cependant souvent utile, lorsque l'on cherche à prouver un nouveau résultat à l'aide de la méthode de priorité, de procéder étapes par étapes, en cherchant à satisfaire un contrat, puis plusieurs simultanément, pour finir par les satisfaire tous, afin de mieux comprendre leurs interactions.

**Corollaire 3.2**
*Il existe un ensemble c. e. $A$ incalculable tel que $A \not\geq_T \emptyset'$.*

Preuve. Soient $A$ et $B$ deux ensembles c. e. tels qu'aucun des deux ne calcule l'autre. Alors, ils sont en particulier tous les deux incalculables. Par ailleurs, si $A$ pouvait calculer l'arrêt, il calculerait aussi $B$ du fait que tout ensemble c. e. est many-one réductible à l'arrêt. ∎

Avant de nous attaquer à des constructions plus élaborées, se basant sur la technique des méthodes de priorités à blessure finie de Friedberg et Muchnik, nous en voyons ici une autre application simple.

**Théorème 3.3**
*Il existe un ensemble c. e. incalculable de degré low.*

Preuve. Nous allons construire un ensemble c. e. $A$ par la méthode de priorité à blessure finie, ainsi qu'une fonction $\Gamma : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ totale calculable stable satisfaisant les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{R}_e : W_e \neq \overline{A} \qquad \mathcal{S}_e : A'(e) = \lim_t \Gamma(e, t).$$

Satisfaire tous les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ assure que l'ensemble $A$ n'est pas calculable, tandis que les contrats $(\mathcal{S}_e)_{e\in\mathbb{N}}$ font en sorte que $A'$ (le problème de l'arrêt relativement à $A$) admet une approximation $\Delta_2^0$, ce qui, par le lemme de limite de Shoenfield (voir le lemme 4-7.2), assure que $A' \leqslant_T \emptyset'$, et donc que $A$ est de degré low.

**Satisfaction d'un contrat** $\mathcal{R}_e$. Voici une stratégie pour satisfaire un contrat $\mathcal{R}_e$, indépendamment des autres contrats. Notre stratégie a trois états : initial (`i`), en exécution (`e`) et terminée (`t`). La stratégie suit les étapes suivantes en fonction de son état.

- ▷ État `i`. Choisir un entier $x_{\mathcal{R}_e} \notin A$, et poser une restreinte sur $x_{\mathcal{R}_e}$.
- ▷ État `e`. Exécuter $\Phi_e(x_{\mathcal{R}_e})$. Si l'exécution s'arrête à l'instant $t$, ajouter $x_{\mathcal{R}_e}$ à $A$, puis passer dans l'état `t`.
- ▷ État `t`. Le processus a terminé son exécution et reste dans cet état.

En supposant que la stratégie n'est blessée qu'un nombre fini de fois, elle pourra avoir deux issues possibles. Issue `p` : elle finira par rester dans l'état `e`, auquel cas, $\Phi_e(x_{\mathcal{R}_e}) \uparrow$ et $x_{\mathcal{R}_e} \notin A$, donc $W_e \neq \overline{A}$. Dans ce cas, le contrat $\mathcal{R}_e$ est dit satisfait passivement. Issue `a` : la stratégie atteindra l'état `t`, et s'arrêtera. Dans ce cas, $\Phi_e(x_{\mathcal{R}_e}) \downarrow$, $x_{\mathcal{R}_e} \in A$, et $\mathcal{R}_e$ est dit satisfait activement.

**Satisfaction d'un contrat $\mathcal{S}_e$.** À l'étape $t$, nous allons définir la valeur de $\Gamma(x,t)$ pour tout $x < t$. La stratégie comporte deux états : en exécution (`e`) et terminée (`t`). Voici la démarche à suivre en fonction de chaque état de la stratégie.

- ▷ État `e`. Exécuter $\Phi_e^A(e)$. Tant que $\Phi_e^{A_t}(e)[t] \uparrow$, maintenir $\Gamma(e,t) = 0$. Si l'exécution s'arrête à l'instant $t$, restreindre l'usage de $\Phi_e^A(e)$ puis passer dans l'état `t`.
- ▷ État `t`. Définir $\Gamma(e,t) = 1$ désormais pour toute nouvelle étape $t$.

Les deux issues possibles de la stratégie sont les suivantes. Issue `p` : elle finira par rester dans l'état `e` d'exécution, auquel cas $\Phi_e^A(e) \uparrow$ et $\lim_t \Gamma(e,t) = 0$. Ainsi, $A'(e) = 0 = \lim_t \Gamma(e,t)$. Dans ce cas, le contrat $\mathcal{S}_e$ est dit satisfait passivement. Issue `a` : la stratégie atteindra l'état `t`, et s'arrêtera. Dans ce cas, $\Phi_e^A(e) \downarrow$ et $\lim_t \Gamma(e,t) = 1$. Donc, $A'(e) = 1 = \lim_t \Gamma(e,t)$, et $\mathcal{S}_e$ est dit satisfait activement.

**Construction**. La construction est globalement la même que celle du théorème 3.1. Les stratégies sont ordonnées $R_0, S_0, R_1, S_1, \ldots$, par priorité décroissante La construction se divise en étapes $t = 0, 1, \ldots$ et chaque étape est elle-même divisée en sous-étapes $s < t$. Initialement, toutes les stratégies sont dans l'état `i`. À l'étape $t \geqslant 0$ et la sous-étape $s < t$, on considère le contrat $\mathcal{R}_e$ si $s = 2e$ et $\mathcal{S}_e$ si $s = 2e+1$. À la sous-étape $s = 2e$, on exécute la stratégie pour $\mathcal{R}_e$ comme décrit ci-dessus en fonction de son état. Lorsqu'elle atteint l'état `t`, toutes les stratégies de priorité inférieure sont blessées et reviennent soit dans l'état `i` dans le cas des stratégies pour des contrats de type $\mathcal{R}$, soit dans l'état `e` dans le cas des stratégies pour des contrats de type $\mathcal{S}$. De la même manière, à la sous-étape $s = 2e + 1$, on exécute la stratégie $\mathcal{S}_e$ comme décrit ci-dessus, en blessant toutes les stratégies de priorité inférieure si l'on arrive à l'état de terminaison `t`.

**Vérification.** L'ensemble $A$ produit est bien c. e., car le processus est calculable, et ne retire aucun élément de $A$ une fois ajouté. On prouve aisément par induction sur $e \in \mathbb{N}$ que les stratégies pour les contrats $\mathcal{R}_e$ et $\mathcal{S}_e$ blessent finiment souvent des stratégies de priorité inférieure. Ainsi, chaque stratégie n'est blessée que finiment souvent, et finira par avoir un comportement limite. Il s'ensuit également que $\lim_t \Gamma(e,t)$ existe et est, par construction, égale à $A'(e)$. Cela conclut la preuve du théorème 3.3. ∎

## 4. Méthode de priorité $\mathbf{\Sigma_2^0}$

Dans les constructions précédentes à l'aide de la méthode de priorité, les blessures finies sont assurées de manière structurelle, c'est-à-dire que dans la structure même de la construction, les stratégies ne posent qu'un nombre

fini de restreintes lorsqu'elles ne sont pas blessées, et sont assurées par construction de n'être blessées qu'un nombre fini de fois. Le nombre de blessures peut même être borné calculatoirement : dans la construction de Friedberg et Muchnik, le $e$-ième processus est blessé au plus $2^e - 1$ fois.

Nous allons maintenant voir une élaboration de la méthode précédente, qui pourrait structurellement résulter en un nombre infini de blessures sur des stratégies, mais des hypothèses sur la construction vont assurer que cette situation n'arrive jamais. Techniquement, il s'agit donc d'une méthode de priorité à blessure finie, car chaque stratégie ne sera blessée qu'un nombre fini de fois. Cependant, cette élaboration peut être vue comme une version dégénérée de la méthode de priorité à blessure infinie, présentée dans la section suivante.

**Théorème 4.1 (Sacks)**
*Pour tout ensemble c. e. incalculable $B$, il existe ensemble c. e. incalculable $A$ qui ne calcule pas $B$.*

Preuve. Nous allons construire un ensemble c. e. $A$ satisfaisant les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ suivants :

$$\mathcal{R}_e : W_e \neq \overline{A} \qquad \mathcal{S}_e : \Phi_e^A = B \Rightarrow B \text{ est calculable.}$$

Satisfaire tous les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ assure que l'ensemble $A$ n'est pas calculable, tandis que les contrats $(\mathcal{S}_e)_{e\in\mathbb{N}}$ font en sorte que $B \not\leqslant_T A$. Le contrat $\mathcal{S}_e$ aurait également pu être noté $\mathcal{W}_e^A \neq \overline{B}$, mais sa formulation actuelle représente mieux la forme de l'argument utilisé pour le satisfaire.

**Satisfaction d'un contrat $\mathcal{R}_e$.** La satisfaction d'un contrat $\mathcal{R}_e$ est exactement la même que pour le théorème 3.3. Nous rappelons les actions de la stratégie en fonction de ses trois états.

- ▷ État `i`. Choisir un entier $x_{\mathcal{R}_e} \notin A$, et poser une restreinte sur $x_{\mathcal{R}_e}$.
- ▷ État `e`. Exécuter $\Phi_e(x_{\mathcal{R}_e})$. Si l'exécution s'arrête à l'instant $t$, ajouter $x_{\mathcal{R}_e}$ à $A$, puis passer dans l'état `t`.
- ▷ État `t`. Le processus est terminé.

Les deux issues de la stratégie sont toujours `p` et `a`.

**Satisfaction d'un contrat $\mathcal{S}_e$.** La stratégie pour satisfaire $\mathcal{S}_e$ est plus complexe et moins intuitive. Elle consiste à essayer de faire coïncider des segments initiaux de $\Phi^A$ et de $B$ de plus en plus longs, de manière calculable, en posant chaque fois des restreintes de plus en plus grandes sur $A$ pour préserver l'usage de $\Phi^A$. À première vue, cette stratégie va donc causer une infinité de blessures aux stratégies de priorité inférieure.

Heureusement, l'ensemble $B$ n'étant pas calculable, la procédure cessera de trouver de nouveaux segments initiaux de coïncidence, et satisfera ainsi le contrat $\mathcal{S}_e$. Plus précisément, la stratégie possède un état initial $\mathtt{i}$, et une infinité d'états $(\mathtt{w}_n)_{n\in\mathbb{N}}$. Pendant l'exécution du processus, nous allons définir une fonction calculable $\Delta : \mathbb{N} \to \{0,1\}$ censée coïncider avec la fonction caractéristique de $B$. Soit $(B_t)_{t\in\mathbb{N}}$ une approximation c. e. de $B$.

- ▷ État $\mathtt{i}$. Définir $\Delta$ comme la fonction de domaine vide, et passer à l'état $\mathtt{w}_0$.
- ▷ État $\mathtt{w}_n$. Attendre une étape $t \geqslant n$, où $\Phi_e^{A_t}[t] \upharpoonright_{n+1} = B_t \upharpoonright_{n+1}$. Si cela arrive, poser alors une restreinte sur l'usage de $\Phi_e^{A_t}[t] \upharpoonright_{n+1}$, définir ensuite $\Delta(n) = B_t(n)$, et passer enfin à l'état $\mathtt{w}_{n+1}$.

Ici, $\Phi_e^{A_t}[t]\upharpoonright_{n+1} = B_t\upharpoonright_{n+1}$ signifie que pour tout $x \leqslant n$, $\Phi_e^{A_t}(x)[t]\downarrow \in \{0,1\}$, et $\Phi_e^{A_t}(x)[t] = 1$ ssi $x \in B_t$ pour $x \leqslant n$. La stratégie ne retourne à l'état $\mathtt{i}$ que lorsqu'elle est blessée. À ce moment-là, la fonction $\Delta$ est également réinitialisée. Cependant, en supposant que la stratégie est blessée un nombre fini de fois, la fonction ne sera réinitialisée que finiment souvent, et sera donc définie de manière calculable.

La stratégie possède une infinité d'issues : pour tout $n$, l'issue $\mathtt{p}_n$ consiste à rester bloqué dans l'état $\mathtt{w}_n$. Si la stratégie ne sort jamais de cet état, alors $\Phi_e^{A_t}[t] \upharpoonright_{n+1} \neq B_t \upharpoonright_{n+1}$ pour tout $t$, donc $\Phi_e^A \neq B$, et le contrat $\mathcal{S}_e$ est satisfait, car la prémisse de l'implication est fausse. La stratégie possède une dernière issue possible, d'ordre infinitaire, qui consiste à passer par tous les états $\mathtt{w}_n$. Nommons cette issue $\infty$. En utilisant l'hypothèse selon laquelle $B$ n'est pas calculable, nous allons maintenant montrer que cette issue ne peut pas arriver.

**Lemme 4.2.** Si l'issue $\infty$ arrive, alors $B$ est calculable. ⋆

Preuve. Soit $e$ le plus petit entier tel que la stratégie pour $\mathcal{S}_e$ passe par tous les états $(\mathtt{w}_n)_{n\in\mathbb{N}}$. Par induction, toutes les stratégies de priorité supérieure sont finiment blessées, et atteindront un état limite où elles ne blesseront plus la stratégie pour $\mathcal{S}_e$. À partir de ce moment, la stratégie pour $\mathcal{S}_e$ passera par tous les états $(\mathtt{w}_n)_{n\in\mathbb{N}}$ successivement. La fonction $\Delta$ définie par le processus est alors totale calculable. Montrons que $\Delta$ est la fonction caractéristique de $B$. Supposons par l'absurde que $\Delta(n) \neq B(n)$, pour un $n \in \mathbb{N}$. Par définition de $\Delta$, $\Delta(n) = B_t(n) = \Phi_e^{A_t}(n)$ pour un $t \in \mathbb{N}$. Comme $B$ est c. e., cette différence vient d'un élément qui apparaît dans $B$, car aucun élément ne peut en sortir : $\Delta(n) = B_t(n) = \Phi_e^{A_t}(n) = 0$ et $n \in B$. Soit $m$ suffisamment grand tel que $n \in B_m$. Par conséquent, à l'état $\mathtt{w}_m$, $\Phi_e^{A_t}(n)[t] = 0 \neq B_t(n)$ pour tout $t \geqslant m$, et $\Phi_e^{A_t}[t] \upharpoonright_{n+1}$ est donc différent de $B_t \upharpoonright_{n+1}$ pour tout $t \geqslant m$, et la stratégie n'atteindra jamais l'état $\mathtt{w}_{m+1}$. Contradiction ! La fonction $\Delta$ est donc la fonction caractéristique de $B$, ce qui prouve que $B$ est calculable. ∎

**Construction**. La construction globale est celle d'une méthode de priorité à blessure finie standard. Les stratégies sont ordonnées par priorité décroissante $R_0, S_0, R_1, S_1, \ldots$ La construction se divise alors en étapes $t = 0, 1, \ldots$, et chaque étape est elle-même divisée en sous-étapes $s < t$. Initialement, toutes les stratégies sont dans l'état `i`. À l'étape $t \geqslant 0$ et la sous-étape $s < t$, on considère le contrat $\mathcal{R}_e$ si $s = 2e$, et $\mathcal{S}_e$ si $s = 2e + 1$. À la sous-étape $s = 2e$, on exécute la stratégie pour $\mathcal{R}_e$ comme décrite ci-dessus en fonction de son état. Lorsqu'elle atteint l'état `e`, toutes les stratégies de priorité inférieure sont blessées, et reviennent dans l'état `i`. De la même manière, à la sous-étape $s = 2e + 1$, on exécute la stratégie pour $\mathcal{S}_e$ suivant les étapes décrites ci-dessus. Chaque fois qu'elle passe à un état suivant $\mathtt{w}_{n+1}$, toutes les stratégies de priorité inférieure sont blessées, et retournent à l'état `i`.

Le lecteur fera la vérification. Cela conclut la preuve du théorème 4.1. ∎

**Stratégie de préservation de Sacks**

La stratégie pour satisfaire $\mathcal{S}_e$ consiste à ne pas essayer de différencier deux ensembles activement, mais au contraire, de préserver des segments initiaux communs de plus en plus longs, pour rendre le processus effectif, puis utiliser l'hypothèse de non-effectivité de l'un des ensembles pour en déduire que ce processus devra échouer. Cette stratégie se retrouve fréquemment dans ce genre de constructions. Elle est parfois appelée *stratégie de préservation de Sacks*, en l'honneur de son auteur.

# 5. Méthode de priorité $\Pi^0_2$ (à blessure infinie)

Nous allons maintenant aborder une nouvelle élaboration de la méthode de priorité, dite à blessure infinie. Comme son nom l'indique, certaines stratégies vont agir infiniment souvent en posant des restreintes de plus en plus grandes, causant des blessures infinies à des stratégies de priorité inférieure. Nous allons donc commencer à avoir des stratégies conditionnelles, adoptant des comportements différents en fonction des issues des stratégies de priorité supérieure, tirant ainsi pleinement parti du formalisme de « l'arbre des stratégies » que nous verrons bientôt.

Notre illustration de la méthode de priorité à blessure infinie concerne l'existence de paires minimales de degrés c. e. Elle améliore le théorème de Friedberg-Muchnik en la combinant avec la stratégie de préservation de Sacks.

**Définition 5.1.** Deux degrés non calculables $\mathbf{a}$ et $\mathbf{b}$ forment une *paire minimale* si leur borne inférieure est $\mathbf{0}$, autrement dit si pour tout ensemble $A$ tel que $A \leqslant_T \mathbf{a}$ et $A \leqslant_T \mathbf{b}$, alors $A$ est calculable. ♢

L'existence de paires minimales de degrés c. e. a été prouvée de manière indépendante par Lachlan [129] et Yates [234].

**Théorème 5.2 (Lachlan 1966, Yates 1966)**
*Il existe une paire minimale de degrés c. e.*

PREUVE. Nous allons construire deux ensembles c. e. $A$ et $B$ satisfaisant les contrats suivants $(\mathcal{R}_e, \mathcal{S}_e, \mathcal{N}_e)_{e\in\mathbb{N}}$ :

$$\mathcal{R}_e : W_e \neq \overline{A} \qquad \mathcal{S}_e : W_e \neq \overline{B} \qquad \mathcal{N}_e : \Phi_e^A = \Phi_e^B \Rightarrow \Phi_e^A \text{ est calculable.}$$

Les contrats $(\mathcal{R}_e)_{e\in\mathbb{N}}$ et $(\mathcal{S}_e)_{e\in\mathbb{N}}$ s'assurent que $A$ et $B$ ne soient pas calculables. Les contrats $(\mathcal{N}_e)_{e\in\mathbb{N}}$ forcent la borne inférieure des degrés de $A$ et $B$ à être $\mathbf{0}$.

**Astuce de Posner**

À première vue, pour imposer que la borne inférieure $\deg_T A$ et $\deg_T B$ est $\mathbf{0}$, on s'attendrait à devoir satisfaire des contrats de la forme $(\mathcal{N}_{i,j})_{i,j\in\mathbb{N}}$ avec

$$\mathcal{N}_{i,j} : \Phi_i^A = \Phi_j^B \Rightarrow \Phi_i^A \text{ est calculable.}$$

Cependant, si $\Phi_i^A = \Phi_j^B$, alors il est possible de créer une nouvelle fonctionnelle $\Phi_e$ qui codera en dur un entier $n$ tel que $A(n) \neq B(n)$, et exécutera $\Phi_i$ ou $\Phi_j$ en fonction de la valeur de son oracle à la position $n$. Ainsi, $\Phi_e^A = \Phi_i^A$ et $\Phi_e^B = \Phi_j^B$. Cette astuce, due à Posner, permet de simplifier les notations.

**Satisfaction d'un contrat $\mathcal{R}_e$ ou $\mathcal{S}_e$.** La satisfaction d'un contrat $\mathcal{R}_e$ ou $\mathcal{S}_e$ est exactement la même que pour le théorème 3.3. Nous rappelons dans le cas de $\mathcal{R}_e$ les actions de la stratégie en fonction de ses trois états :

- ▷ État `i`. Choisir un entier $x_{\mathcal{R}_e} \notin A$, et poser une restreinte sur $x_{\mathcal{R}_e}$.
- ▷ État `e`. Exécuter $\Phi_e(x_{\mathcal{R}_e})$. Si l'exécution s'arrête à l'instant $t$, ajouter $x_{\mathcal{R}_e}$ à $A$, puis passer dans l'état `t`.
- ▷ État `t`. Le processus est terminé.

Jusqu'ici, nous avons considéré que cette stratégie avait deux issues, en fonction de l'état dans lequel elle se stabilise. Ces deux issues sont de même nature finitaire, en ce qu'elles ne posent qu'un nombre fini de restreintes lorsqu'elles sont blessées finiment souvent. Nous les considérerons donc comme une seule issue `f`.

**Satisfaction d'un contrat $\mathcal{N}_e$.** La satisfaction d'un contrat $\mathcal{N}_e$ va suivre la stratégie de préservation de Sacks pour préserver des segments initiaux communs à $\Phi_e^A$ et $\Phi_e^B$ de plus en plus longs.

Cependant, contrairement au théorème 4.1, le processus ne va pas nécessairement échouer au bout d'un nombre fini d'étapes, car rien dans les hypothèses n'empêche cette égalité. Nous sommes donc dans un cas où l'issue infinitaire va pouvoir se produire, avec des restreintes de plus en plus longues, résultant en une blessure infinie. Comme dans le cas du théorème 4.1, la stratégie possède un état initial $\mathtt{i}$, et une infinité d'états $(\mathtt{w}_n)_{n\in\mathbb{N}}$. Dans ce qui suit, nous appellerons *usage* de $\Phi_e^{A_t}[t]\restriction_{n+1}$ le maximum des usages de $\{\Phi_e^{A_t}(x)[t] : x \leqslant n\}$. Pendant l'exécution de la stratégie, nous allons définir une fonction calculable $\Delta : \mathbb{N} \to \{0,1\}$ telle que si $\Phi_e^A$ et $\Phi_e^B$ sont totaux et égaux, alors ils sont tous les deux égaux à $\Delta$.

- ▷ État $\mathtt{i}$. Définir $\Delta$ comme la fonction de domaine vide, et passer à l'état $\mathtt{w}_0$.
- ▷ État $\mathtt{w}_n$. Attendre une étape $t \geqslant n$ où $\Phi_e^{A_t}[t]\restriction_{n+1} = \Phi_e^{B_t}[t]\restriction_{n+1}$. Si cela arrive, relâcher sa restreinte précédente, et poser une restreinte sur l'usage de $\Phi_e^{A_t}[t]\restriction_{n+1}$ si $n$ est pair, et sur l'usage de $\Phi_e^{B_t}[t]\restriction_{n+1}$ si $n$ est impair. Ensuite, définir $\Delta(n) = \Phi_e^{A_t}(n)[t]$, et passer à l'état $\mathtt{w}_{n+1}$.

La stratégie a deux issues possibles. Issue $\mathtt{f}$ (finitaire) : elle se retrouve bloquée dans l'état $\mathtt{w}_n$ pour un $n$ donné ; dans ce cas, soit $\Phi_e^A$ ou $\Phi_e^B$ est partiel, soit $\Phi_e^A \neq \Phi_e^B$. Issue $\infty$ (infinitaire) : la stratégie passe par tous les états $(\mathtt{w}_n)_{n\in\mathbb{N}}$ ; dans ce cas, les deux ensembles coïncident, et infiniment souvent, la restreinte change de côté. Il nous reste encore à établir les égalités $\Delta = \Phi_e^A = \Phi_e^B$ pour en déduire que cet ensemble est calculable. L'idée sous-jacente de la preuve est très simple, mais elle est un peu lourde à formaliser. Nous allons donc l'illustrer par une figure (voir la figure 5.3) avant de prouver formellement le résultat à travers le lemme 5.4. Quelle que soit l'issue, le contrat $\mathcal{N}_e$ est donc satisfait.

**Remarque**

Le choix du côté de la restreinte ($A$ si la stratégie passe d'un état $\mathtt{w}_n$ à $\mathtt{w}_{n+1}$ avec $n$ pair, et $B$ si $n$ est impair) n'intervient pas dans la preuve de la validité d'une stratégie $\mathcal{N}_e$ indépendamment des autres. On aurait aussi bien pu garder toujours le même côté, ou bien même poser une restreinte des deux côtés, ce qui aurait considérablement simplifié la preuve de validité. Cependant, cette alternance de côtés devient nécessaire lorsque l'on cherche à satisfaire un contrat de type $\mathcal{R}$ ou $\mathcal{S}$ sous une stratégie pour $\mathcal{N}_e$, comme nous le verrons par la suite.

**Lemme 5.4.** Si l'issue $\infty$ arrive, alors $\Delta = \Phi_e^A = \Phi_e^B$. ⋆

Preuve. Soit $P(n,s)$ la proposition « Soit (1) $\Delta\restriction_{n+1} = \Phi_e^{A_s}[s]\restriction_{n+1}$ avec une restreinte sur son usage, soit (2) $\Delta\restriction_{n+1} = \Phi_e^{B_s}[s]\restriction_{n+1}$ avec une restreinte sur son usage. »

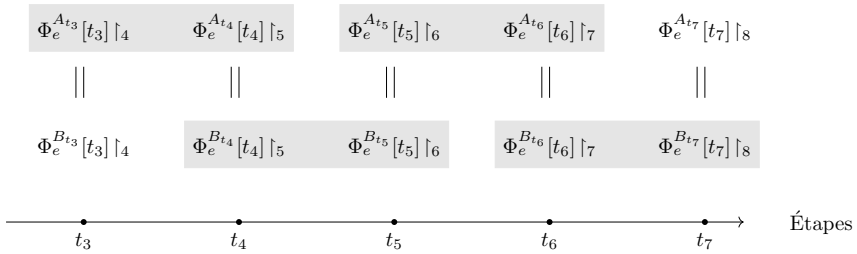



FIGURE 5.3 – *Soit $t_n$ l'étape de passage à l'état $\mathtt{w}_{n+1}$. Le rectangle gris entre l'étape $t_3$ et $t_4$ signifie que l'usage de $\Phi_e^{A_{t_3}}[t_3]\restriction_4$ est restreint, et donc préservé jusqu'à l'étape $t_4$. Ainsi, $\Phi_e^{A_{t_3}}[t_3]\restriction_4 = \Phi_e^{A_{t_4}}[t_4]\restriction_4$. À chaque étape $t_n$, les segments initiaux de longueur $n+1$ des deux fonctionnelles sont égaux. Ainsi, $\Phi_e^{A_{t_4}}[t_4]\restriction_5 = \Phi_e^{B_{t_4}}[t_4]\restriction_5$. Nous avons donc*

$$\Phi_e^{B_{t_3}}[t_3]\restriction_4 = \Phi_e^{A_{t_3}}[t_3]\restriction_4 = \Phi_e^{A_{t_4}}[t_4]\restriction_4 = \Phi_e^{B_{t_4}}[t_4]\restriction_4 .$$

*Même si l'usage de $\Phi_e^{B_{t_3}}[t_3]\restriction_4$ peut être différent de celui de $\Phi_e^{B_{t_4}}[t_4]\restriction_4$, car aucune restreinte n'est posée sur $B$ entre les étapes $t_3$ et $t_4$, les quatre premières valeurs de sortie de la fonctionnelle sont préservées.*

Pour tout $n$, soit $t_n$ l'étape à laquelle la stratégie passe dans l'état $\mathtt{w}_{n+1}$. Nous allons montrer par induction sur les entiers $n$ et $s$ que pour tout $n \geqslant 0$ et $s \geqslant t_n$, la proposition $P(n,s)$ est vraie. Par convention, $t_{-1} = 0$, et pour tout $s \geqslant t_{-1}$, $P(-1,s)$ est vraie.

Soit $n \geqslant 0$. Montrons que si pour tout $s \geqslant t_{n-1}$, $P(n-1,s)$ est vraie, alors $P(n,t_n)$ est vraie. À l'étape $t_n$, $\Delta(n) = \Phi_e^{A_{t_n}}(n)[t_n] = \Phi_e^{B_{t_n}}(n)[t_n]$, et $\Phi_e^{A_{t_n}}[t_n]\restriction_{n+1} = \Phi_e^{B_{t_n}}[t_n]\restriction_{n+1}$. Comme $t_n \geqslant t_{n-1}$, par hypothèse d'induction, $P(n-1,t_n)$ est vraie, de sorte que l'on a soit $\Delta\restriction_n = \Phi_e^{A_{t_n}}[t_n]\restriction_n$, soit $\Delta\restriction_n = \Phi_e^{B_{t_n}}[t_n]\restriction_n$. Par suite, $\Delta\restriction_{n+1} = \Phi_e^{A_{t_n}}[t_n]\restriction_{n+1} = \Phi_e^{B_{t_n}}[t_n]\restriction_{n+1}$. À cette étape-ci, la stratégie pose une restreinte sur $\Phi_e^{A_{t_n}}[t_n]\restriction_{n+1}$ ou bien sur $\Phi_e^{B_{t_n}}[t_n]\restriction_{n+1}$, donc $P(n,t_n)$ est vraie.

Soit $s > t_n$. Montrons que si $P(n,s-1)$ est vraie, alors $P(n,s)$ est vraie. Si la stratégie ne change pas d'état à l'étape $s$, alors elle garde sa restreinte, et par la propriété de l'usage, $P(n,s)$ reste vraie. Si la stratégie passe à un état $\mathtt{w}_{p+1}$, alors par définition, $\Phi_e^{A_s}[s]\restriction_{p+1} = \Phi_e^{B_s}[s]\restriction_{p+1}$, or $p \geqslant n$, donc $\Phi_e^{A_s}[s]\restriction_{n+1} = \Phi_e^{B_s}[s]\restriction_{n+1}$.
Par $P(n,s-1)$, soit $\Delta\restriction_{n+1} = \Phi_e^{A_{s-1}}[s-1]\restriction_{n+1}$ avec une restreinte sur son usage, soit (2) $\Delta\restriction_{n+1} = \Phi_e^{B_{s-1}}(n)[s-1]\restriction_{n+1}$ avec une restreinte sur son

usage. Par la restreinte à l'étape $s-1$ et la propriété de l'usage,

$$\text{soit} \quad \Delta\restriction_{n+1} = \Phi_e^{A_s}[s]\restriction_{n+1}, \qquad \text{soit} \quad \Delta\restriction_{n+1} = \Phi_e^{B_s}[s]\restriction_{n+1}\ .$$

Il s'ensuit que $\Delta\restriction_{n+1} = \Phi_e^{A_s}[s]\restriction_{n+1} = \Phi_e^{B_s}[s]\restriction_{n+1}$. La stratégie pose une restreinte sur l'usage de $\Phi_e^{A_s}[s]\restriction_{p+1}$ ou $\Phi_e^{B_s}[s]\restriction_{p+1}$, donc $P(n,s)$ est vraie. Cela termine la preuve du lemme. ∎

Notons que contrairement au théorème 4.1, l'issue infinitaire va vraiment se produire avec la stratégie pour $\mathcal{N}_e$. Elle ne se combine donc pas aussi bien avec les stratégies pour $\mathcal{R}_d$ et $\mathcal{S}_d$. En effet, lorsque cette issue arrive, la stratégie pour $\mathcal{N}_e$ va poser des restreintes de plus en plus longues, causant une blessure infinie. Nous allons donc devoir adapter la construction pour permettre aux autres contrats d'être satisfaits.

**Étapes d'exécution**

Il n'est pas nécessaire d'exécuter les stratégies pour $\mathcal{R}_e$, $\mathcal{S}_e$ et $\mathcal{N}_e$ à chaque étape pour satisfaire leurs contraintes respectives. Il suffit de les exécuter chacune pendant un nombre infini d'étapes, tout en maintenant leurs restreintes pendant les étapes intermédiaires.
Prenons l'exemple de la stratégie pour $\mathcal{R}_e$. Si elle n'est exécutée qu'aux temps $t_0 < t_1 < \ldots$, il se peut qu'elle « manque » la première étape $t$ entre $t_0$ et $t_1$ où $\Phi_e(x_{\mathcal{R}_e})[t]$ s'arrête, ce qui l'empêche de passer dans l'état `t` à cette étape. Cependant, à l'étape $t_1$, $\Phi_e(x_{\mathcal{R}_e})[t_1]$ s'arrêtera également, et le passage à l'état `t` aura quand même lieu. Le comportement limite de la stratégie pour $\mathcal{R}_e$ ne dépend donc pas du choix des étapes.
Le cas de la stratégie pour $\mathcal{N}_e$ est un peu plus subtil. Il se peut que la stratégie par défaut pour $\mathcal{N}_e$ ait une issue infinitaire, mais que lorsqu'elle n'est exécutée qu'aux temps $t_0 < t_1 < \ldots$, on ait

$$\Phi_e^{A_t}[t_i]\restriction_{n+1} \neq \Phi_e^{B_t}[t_i]\restriction_{n+1},$$

ce qui fait que la stratégie ne passera jamais à l'état $\mathtt{w}_{n+1}$, ce qui correspond à l'issue finitaire. Heureusement, même dans ce cas, $\mathcal{N}_e$ sera satisfait, tant que l'énumération des étapes $(t_i)_{i\in\mathbb{N}}$ est calculable pour que la fonction $\Delta$ le soit également.

**Satisfaction d'un contrat $\mathcal{R}_d$ ou $\mathcal{S}_d$ sous $\mathcal{N}_e$.** Supposons que l'on veuille satisfaire un contrat $\mathcal{R}_d$ sous la stratégie pour $\mathcal{N}_e$, c'est-à-dire avec la stratégie pour $\mathcal{N}_e$ de priorité supérieure à celle pour $\mathcal{R}_d$. Plusieurs solutions se présentent, en fonction de l'issue de la stratégie pour $\mathcal{N}_e$.

- ▷ Issue `f` (finitaire). Dans ce cas, il suffit d'utiliser la stratégie standard pour $\mathcal{R}_d$ présentée plus haut. En effet, la stratégie pour $\mathcal{N}_e$ posera un

nombre fini de restreintes, ce qui fait que la stratégie pour $\mathcal{R}_d$ sera blessée et réinitialisée un nombre fini de fois avant d'être satisfaite. Cette stratégie ne fonctionne pas si l'issue de la stratégie pour $\mathcal{N}_e$ est infinitaire (issue $\infty$). En effet, dans ce cas, la stratégie pour $\mathcal{R}_d$ sera blessée infiniment souvent, et pourrait ne jamais satisfaire $\mathcal{R}_d$.

- ▷ Issue $\infty$ (infinitaire). Remarquons que dans ce cas, la restreinte posée par la stratégie pour $\mathcal{N}_e$ alternera infiniment souvent de côté, et libérera donc l'autre côté, permettant à la stratégie pour $\mathcal{R}_d$ d'être satisfaite. Nous allons donc n'exécuter la stratégie pour $\mathcal{R}_d$ qu'aux étapes où la restreinte de la stratégie pour $\mathcal{N}_e$ est retirée du côté $A$. L'issue de la stratégie pour $\mathcal{N}_e$ étant infinitaire, la stratégie pour $\mathcal{R}_d$ va être exécutée pendant un nombre infini d'étapes, et comme expliqué plus haut, un sous-ensemble infini d'étapes est suffisant pour satisfaire $\mathcal{R}_d$. Cette stratégie pour $\mathcal{R}_d$ ne fonctionne cependant pas si l'issue de la stratégie pour $\mathcal{N}_e$ est finitaire, car il se peut qu'elle ne lève jamais sa restreinte et que la stratégie pour $\mathcal{R}_d$ attende donc pour toujours.

Nous avons donc deux stratégies différentes pour satisfaire $\mathcal{R}_d$ sous $\mathcal{N}_e$, en fonction de l'issue de la stratégie pour $\mathcal{N}_e$. Cette analyse de cas pose une difficulté : pour produire un ensemble c. e., il faut que la construction soit un processus calculable, or l'issue de $\mathcal{N}_e$ ne peut pas être décidée en un temps fini. Nous ne pouvons donc pas savoir quelle stratégie choisir pour $\mathcal{R}_d$. La solution consiste à lancer les deux stratégies pour $\mathcal{N}_e$ en parallèle, chacune faisant une supposition sur l'issue. Celle faisant la bonne supposition pourra alors satisfaire $\mathcal{R}_d$ sous $\mathcal{N}_e$. Nous allons donc nous retrouver avec un arbre de stratégies, induit par les analyses de cas successives sur les issues des stratégies de type $\mathcal{N}$. Nous détaillerons plus bas cette structure arborescente.

**Satisfaction d'un contrat $\mathcal{N}_d$ sous $\mathcal{N}_e$.** Les contrats de même nature sont souvent faciles à satisfaire simultanément, car leurs stratégies n'entrent généralement pas en conflit. Dans le cas de la satisfaction d'un contrat $\mathcal{N}_d$ sous $\mathcal{N}_e$, la difficulté n'est pas de satisfaire ces deux contrats simultanément, mais de laisser ensuite la place à un contrat de type $\mathcal{R}$ ou $\mathcal{S}$ d'être satisfait sous $\mathcal{N}_d$. En effet, dans le pire des cas, les stratégies pour $\mathcal{N}_e$ et $\mathcal{N}_d$ seront toutes les deux infinitaires, et à chaque fois que la stratégie pour $\mathcal{N}_e$ libérera ses restreintes du côté $A$, la stratégie pour $\mathcal{N}_d$ posera les siennes, ce qui fait que le côté $A$ aura à tout moment des restreintes de plus en plus grandes, ne permettant pas aux stratégies de type $\mathcal{R}$ d'être satisfaites en dessous. La solution consiste à « synchroniser » les stratégies pour $\mathcal{N}_d$ et $\mathcal{N}_e$. Plus précisément, la stratégie pour $\mathcal{N}_d$ va être définie par analyse de cas en fonction de l'issue de $\mathcal{N}_e$.

- ▷ Issue $\mathtt{f}$ (finitaire). Dans ce cas, la stratégie pour $\mathcal{N}_d$ est la stratégie standard présentée plus haut. En effet, les restreintes de la stratégie pour $\mathcal{N}_e$ seront finitaires, et les autres contrats auront la possibilité d'être satisfaits sous $\mathcal{N}_d$ en étant blessés finiment souvent par la stratégie de $\mathcal{N}_e$.
- ▷ Issue $\infty$ (infinitaire). Modifions la stratégie pour $\mathcal{N}_d$, comme suit. Cette stratégie ne sera exécutée que pendant des étapes où la stratégie pour $\mathcal{N}_e$ change ses restreintes de côté, autrement dit passe de l'état $\mathtt{w}_n$ à $\mathtt{w}_{n+1}$. Cela permet de s'assurer que lorsque la stratégie pour $\mathcal{N}_d$ change ses restreintes de côté, à la même étape, la stratégie pour $\mathcal{N}_e$ changera les siennes de côté. Enfin, il faut s'assurer que ces changements aillent du même côté. Pour cela, si la stratégie pour $\mathcal{N}_d$ est dans un état $\mathtt{w}_n$ avec $n$ un entier pair, autrement dit si ses restreintes sont du côté $B$, la stratégie pour $\mathcal{N}_d$ ne sera exécutée que pendant les étapes où la stratégie pour $\mathcal{N}_e$ passe d'un état $\mathtt{w}_m$ à $\mathtt{w}_{m+1}$ avec $m$ pair, pour que les deux stratégies mettent leur restreintes sur le côté $A$ simultanément. De la même manière, si la stratégie pour $\mathcal{N}_d$ est dans un état $\mathtt{w}_n$ avec $n$ un entier impair, elle s'exécutera pendant les étapes où la stratégie pour $\mathcal{N}_e$ passe d'un état $\mathtt{w}_m$ à $\mathtt{w}_{m+1}$ avec $m$ impair.

Comme dans le cas de la satisfaction d'un contrat $\mathcal{R}_d$ sous un contrat $\mathcal{N}_e$, nous avons donc des stratégies pour $\mathcal{N}_d$ différentes pour chaque issue de la stratégie pour $\mathcal{N}_e$. Nous allons donc lancer deux stratégies en parallèle, chacune supposant que son hypothèse est la bonne. Nous allons maintenant décrire l'arbre des stratégies.

**Arbre des stratégies**. Les contrats sont énumérés de la manière suivante.

$$\mathcal{N}_0, \mathcal{R}_0, \mathcal{S}_0, \mathcal{N}_1, \mathcal{R}_1, \mathcal{S}_1, \ldots$$

Dans les constructions précédentes, chaque contrat se voyait attribuer une unique stratégie, et l'ordre de priorité des stratégies suivait l'énumération des contrats. Nous avons maintenant une structure arborescente de stratégies en fonction des issues des stratégies précédentes, comme ci-après : le contrat $\mathcal{N}_0$ possède une unique stratégie $N_\epsilon$ (où $\epsilon$ est la chaîne vide). Le contrat $\mathcal{R}_0$ admet deux stratégies $R_\infty$ et $R_\mathtt{f}$, en fonction des issues $\infty$ et $\mathtt{f}$ de la stratégie $\mathcal{N}_0$. Le contrat $\mathcal{S}_0$ possède également deux stratégies $S_{\infty\mathtt{f}}$ et $S_{\mathtt{ff}}$, en fonction des issues des stratégies plus hautes. Par exemple, $S_{\infty\mathtt{f}}$ doit satisfaire le contrat $\mathcal{S}_0$ sous l'hypothèse que l'issue de $N_\epsilon$ est $\infty$ et l'issue de $R_\infty$ est $\mathtt{f}$.

De manière générale, nous pouvons définir un arbre $\mathcal{T} \subseteq \{\mathtt{f}, \infty\}^{<\mathbb{N}}$ de chaînes dans l'alphabet $\{\mathtt{f}, \infty\}$, induit par la relation de préfixe, tel que pour tout $\sigma \in \mathcal{T}$ et $i < |\sigma|$ tel que $i \not\equiv 0 \mod 3$, $\sigma(i) = \mathtt{f}$. À chaque chaîne $\sigma \in \mathcal{T}$, si $|\sigma| = 3e$ on associe une stratégie $N_\sigma$ pour $\mathcal{N}_e$, si $|\sigma| = 3e+1$ on associe une stratégie $R_\sigma$ pour $\mathcal{R}_e$, et si $|\sigma| = 3e + 2$ on associe une

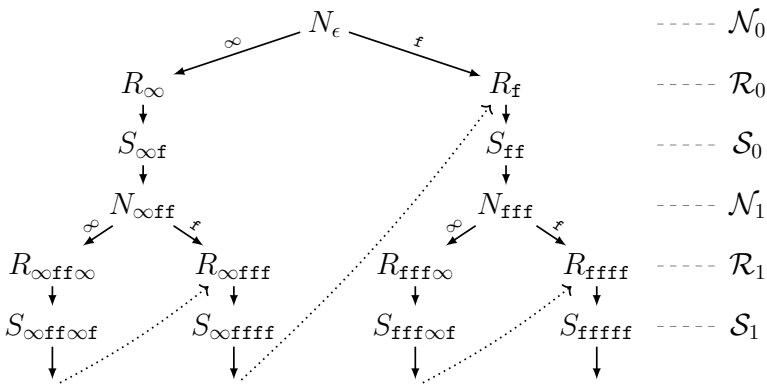



FIGURE 5.5 – *Arbre des stratégies pour la construction d'une paire minimale. Les flèches pleines indiquent la structure d'arbre, mais également induisent un ordre partiel de priorité entre les stratégies. Les flèches en pointillés permettent de linéariser l'ordre partiel pour obtenir un ordre total entre les stratégies. Par exemple, toutes les stratégies du sous-arbre gauche de la stratégie $N_{\infty\mathtt{ff}}$ sont plus faibles que $N_\epsilon$, $R_\infty$, $S_{\infty\mathtt{f}}$ et $N_{\infty\mathtt{ff}}$, mais sont plus fortes que toutes les autres stratégies.*

stratégie $S_\sigma$ pour $\mathcal{S}_e$ (voir la figure 5.5). Pour simplifier les notations, nous noterons $C_\sigma$ la stratégie dont l'indice est $\sigma$.

**Construction.** Au début de la construction, chaque stratégie est dans l'état initial `i`. À chaque étape $t$, nous allons définir un *nœud courant*, qui est une chaîne $\sigma_t \in \mathcal{T}$ de longueur $t$, représentant une approximation des issues à la fin de l'étape $t$. Plus précisément, $\sigma_t(i)$ représente l'approximation à l'étape $t$ de l'issue de la stratégie $C_{\sigma_t\restriction_i}$. Le nœud courant à l'étape $t$ est défini par induction sur les sous-étapes $s < t$ comme suit.
Au début de la sous-étape $i$, on suppose $\sigma_t \restriction_i$ défini. On exécute alors la stratégie $C_{\sigma_t\restriction_i}$ au temps $t$ comme décrit ci-dessus. Rappelons qu'il se peut que l'on n'exécute pas la stratégie $C_{\sigma_t\restriction_i}$ pour des raisons de synchronisation avec les stratégies pour les contrats de type $\mathcal{N}$. Dans ce cas, la stratégie reste dans son état et garde ses contraintes inchangées. À la fin de la sous-étape $t$, on définit $\sigma_t(i) = \infty$ si $i \equiv 0 \mod 3$, et la stratégie $N_{\sigma_t\restriction_i}$ a changé d'état à l'étape $t$. Dans tous les autres cas, $\sigma_t(i) = \mathtt{f}$, car les stratégies de type $\mathcal{R}$ et $\mathcal{S}$ n'ont que l'issue finitaire possible, et si $i \equiv 0 \mod 3$, et $N_{\sigma_t\restriction_i}$ ne vient pas de changer d'état, nous supposons que son issue est finitaire.

**Vrai chemin**. Nous pouvons maintenant définir le *vrai chemin* de $\mathcal{T}$, qui est le chemin le long duquel les issues sont les vraies. Plus précisément, le vrai chemin de $\mathcal{T}$ est la suite infinie $P \in \Lambda^{\mathbb{N}}$ (avec $\Lambda = \{\infty, f\}$, où $\infty$ est

plus petit que $f$) définie inductivement sur $i$ par

$$P(i) = \liminf\{o \in \Lambda : \exists^\infty t \ (P\restriction_i)^\frown o \preceq \sigma_t\}.$$

Intuitivement, les stratégies le long du vrai chemin vont être celles qui feront les bonnes hypothèses sur les issues des stratégies précédentes. Elles seront blessées finiment souvent par une stratégie de plus grande priorité, et réussiront à satisfaire leur contrat. Nous allons maintenant définir un ordre de priorité sur les stratégies pour que les stratégies le long du vrai chemin soient finiment blessées.

**Ordre de priorité**. Soit $\Lambda = \{\infty, \mathtt{f}\}$ l'ensemble des issues possibles. Munissons cet ensemble d'un ordre de priorité $<_p$ en considérant que $\infty <_p \mathtt{f}$, ce qui signifie que l'issue $\infty$ est prioritaire par rapport à $\mathtt{f}$. Cet ordre va induire un ordre total $(\mathcal{T}, <_p)$ (et donc un ordre total de priorité sur les stratégies) comme suit : $\sigma <_p \tau$ si $\sigma \preceq \tau$, ou si $i$ est la première position où les deux chaînes diffèrent, et $\sigma(i) <_p \tau(i)$. Un exemple visuel de cet ordre est donné dans la figure 5.5. Notons que, contrairement aux méthodes de priorité à blessure finie, les stratégies sont généralement en dessous d'une infinité de stratégies de priorité supérieure. Par exemple, la stratégie $R_{\infty\mathtt{fff}}$ est en dessous de $N_\epsilon$, $R_\infty$, $S_{\infty\mathtt{f}}$, $N_{\infty\mathtt{ff}}$, $R_{\infty\mathtt{ff}\infty}$, $S_{\infty\mathtt{ff}\infty\mathtt{f}}, \ldots$

**Remarque**

On serait tenté de définir un ordre de priorité tel que chaque stratégie n'ait qu'un nombre fini de stratégies prioritaires, par exemple en définissant les priorités par un parcours en largeur des nœuds de l'arbre. Le problème suivant se pose cependant.
Supposons que la stratégie $N_\epsilon$ ait pour issue $\infty$. La stratégie $R_\mathtt{f}$ faisant la mauvaise hypothèse selon laquelle l'issue de $N_\epsilon$ a une issue finitaire, elle sera blessée et réinitialisée à chaque changement d'état de $N_\epsilon$. Elle posera donc potentiellement un nombre infini de restreintes, et empêchera donc toutes les stratégies de priorité inférieure de fonctionner normalement. Il est donc essentiel que toutes les stratégies le long du vrai chemin de l'arbre soient prioritaires par rapport à $R_\mathtt{f}$, afin d'ignorer ses restreintes. La stratégie pour $R_\mathtt{f}$ doit donc être sous une infinité de stratégies prioritaires.

**Vérification**. Notons tout d'abord que, par définition, les stratégies le long du vrai chemin $P$ (stratégies $C_\sigma$ pour $\sigma \prec P$) sont exécutées à une infinité d'étapes. Bien qu'une stratégie soit en général en dessous d'une infinité de stratégies, nous allons montrer que les stratégies le long du vrai chemin sont blessées finiment souvent, ce qui est la condition nécessaire pour les satisfaire. Le lemme suivant est une propriété que l'on attend généralement d'un argument de priorité $\Pi^0_2$.

**Lemme 5.6.** Soit $P$ le vrai chemin, et soit $\alpha \prec P$. Alors, $\alpha \leqslant_p \sigma_t$ pour un nombre co-fini d'étapes $t$. ⋆

Preuve. Par induction sur la longueur $n$ de $\alpha$. Si $n = 0$, alors $\alpha = \epsilon$, et par définition, $\epsilon \leqslant_p \sigma_t$ pour tout $t$. Soit $n > 0$. Par hypothèse d'induction, il existe un seuil $t_0$ tel que $\alpha\restriction_{n-1} \leqslant_p \sigma_t$ pour tout $t \geqslant t_0$. Soit

$$S = \{t \geqslant t_0 : \sigma_t <_p \alpha\}.$$

Supposons par l'absurde que $S$ est infini.
Notons que pour tout $t \in S$, comme $\alpha \restriction_{n-1} \leqslant_p \sigma_t$ et $\sigma_t <_p \alpha$, on a forcément $\alpha\restriction_{n-1} \preceq \sigma_t$ avec $\sigma_t(n-1) <_p \alpha(n-1)$. Autrement dit, $\sigma_t(n-1) = \infty$ et $\alpha(n-1) = \mathtt{f}$ et $\forall t \in S\ (\alpha \restriction_{n-1})^\frown \infty \preceq \sigma_t$. Ainsi, $(\alpha \restriction_{n-1})^\frown \infty \preceq P$, contredisant l'hypothèse $\alpha \preceq P$. Cela conclut la preuve du lemme. ∎

Seules les stratégies le long de $\sigma_t$ sont exécutées à l'étape $t$. Ainsi, pour toute stratégie $C_\alpha$ le long du vrai chemin $P$, il existe un seuil $t_0$ après lequel seules les stratégies de plus faible priorité ou des stratégies le long du vrai chemin vont être exécutées. Nous pouvons donc prouver par induction sur $n$ que la stratégie $C_{P\restriction_n}$ ne sera blessée que finiment souvent, et aura l'issue $P(n)$. Tout contrat étant représenté par une stratégie le long du vrai chemin, les contrats seront tous satisfaits. Cela conclut la preuve du théorème 5.2. ∎

### Degrés cappables

Nous avons vu deux manières de créer des degrés Turing incomparables. La première (voir la proposition 4-8.1) consiste à créer deux ensembles simultanément, tandis que la seconde (voir la proposition 4-8.2) part d'un ensemble non calculable, et crée un deuxième ensemble de degré incomparable avec le premier. Il est naturel de se demander s'il est possible, dans le cas des paires minimales de degrés c. e., de partir d'un ensemble c. e. non calculable arbitraire, et le compléter avec un autre ensemble c. e. pour former une paire minimale. Ce n'est pas le cas, comme l'ont prouvé Yates [234] et Lachlan [129].

**Définition 5.7.** Un degré c. e. $\mathbf{a} > \mathbf{0}$ est dit *cappable* s'il existe un degré $\mathbf{b} > \mathbf{0}$ tel que $\mathbf{a}$ et $\mathbf{b}$ forment une paire minimale. Sinon, $\mathbf{a}$ est dit *non cappable*. ♢

Notons que le terme « cappable » vient de l'anglais « cap » et « able », et n'a en particulier rien à avoir avec l'adjectif « capable ». Ambos-Pies et al. [5] ont obtenu une surprenante caractérisation des degrés non cappables, à l'aide des ensembles promptement simples.

**Définition 5.8.** Un ensemble co-infini c. e. $A$ est *promptement simple* s'il existe une énumération c. e. calculable $A_0 \subseteq A_1 \subseteq \dots$ et une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ telles que

$$W_e \text{ infini } \Rightarrow \exists^\infty x, s\ (x \in W_e[s] \setminus W_e[s-1] \wedge x \in A_{f(s)}). \qquad \diamondsuit$$

Autrement dit, un ensemble co-infini $A$ est promptement simple s'il est non seulement co-immune, mais plus encore, cette co-immunité doit être réalisée en faisant infiniment souvent entrer des éléments dans $A$ peu de temps après qu'ils apparaissent dans $W_e$.

Un degré c. e. est *promptement simple* s'il contient un ensemble promptement simple.

**Théorème 5.9 (Ambos-Spies, Jockusch, Shore, et Soare [5])**
*Les degrés non cappables sont précisément les degrés promptement simples.*

En particulier, la preuve montrant qu'un degré est cappable s'il n'est pas promptement simple est obtenue avec une variation du théorème 5.2.

Chapitre 14

# Structure des degrés Turing

L'étude des degrés Turing a été menée conjointement avec celle de sa *structure*, en tant qu'ordre partiel. Gerald E. Sacks est sans aucun doute l'un des principaux protagonistes de cette aventure.

Gerald Sacks, 1933–2019

Sacks entame dans sa jeunesse des études d'ingénieur à l'université de Cornell, qu'il interrompt à mi-parcours pour s'engager trois ans durant dans l'armée. C'est à ce moment qu'il met la main sur une copie de *Introduction to Metamathematics* de Kleene, qui le passionne [35]. À son retour à la vie civile, il oriente alors la suite de ses études vers les mathématiques. Barkley Rosser, éminent logicien qui fut étudiant avec Kleene auprès de Church, accepte de le prendre comme étudiant en thèse. Sacks deviendra dans les années soixante l'un des pionniers de la calculabilité moderne. Il participera notamment comme nous le verrons aux premières études sur la structure des degrés Turing, et outre son travail, il deviendra célèbre pour le grand nombre de ses étudiants qui deviendront des logiciens de premier plan, parmi lesquels on peut citer Harvey et Sy Friedman, Léo Harrington, Richard Shore, Théodore Slaman et Stephen Simpson. Nous verrons que les trois derniers

membres de cette liste ont pris une part très active dans l'étude de la structure des degrés Turing.

Posons sans plus tarder le vocabulaire que nous utiliserons.

**Notation**

On notera $(\boldsymbol{\mathcal{D}}, \leqslant)$ la structure d'ordre partiel des degrés Turing.

On écrira $\mathbf{a} \leqslant \mathbf{b}$ pour deux degrés $\mathbf{a}, \mathbf{b} \in \boldsymbol{\mathcal{D}}$ si $A \leqslant_T B$ pour tout élément $A \in \mathbf{a}$ et tout élément $B \in \mathbf{b}$, et l'on écrira $\mathbf{a} < \mathbf{b}$ si $\mathbf{a} \leqslant \mathbf{b}$ et $\mathbf{b} \not\leqslant \mathbf{a}$. Afin d'être tout à fait clairs, rappelons le vocabulaire d'usage des ordres partiels : étant donné un ensemble partiellement ordonné $A$ et un sous-ensemble $B \subseteq A$, un *majorant* (resp. *minorant*) de $B$ est un élément de $A$ plus grand (resp. plus petit) que tous les éléments de $B$. Un majorant (resp. minorant) de $B$ est *minimal* (resp. *maximal*) si aucun autre majorant de $B$ n'est plus petit que lui (resp. aucun autre minorant de $B$ n'est plus grand que lui). Enfin, un majorant (resp. minorant) de $B$ est une *borne supérieure* (resp. *borne inférieure*) si elle est plus petite que tout majorant de $B$ (resp. plus grande que tous les minorants de $B$). On montre sans peine qu'une borne supérieure (resp. inférieure) quand elle existe est unique.

La littérature sur le sujet étant exclusivement en anglais, nous mettons l'accent sur le fait que l'anglais *upper bound* et *lower bound* ne correspond pas au français « borne supérieure » et « borne inférieure », mais plutôt à majorant et minorant. L'anglais utilise à la place *least upper bound* et *greatest lower bound* pour borne supérieure et borne inférieure.

## 1. Degrés minimaux

Un des tout premiers résultats obtenus sur la structure des degrés Turing concerne les segments initiaux de $\boldsymbol{\mathcal{D}}$, et notamment l'existence de degrés dits *minimaux*.

**Définition 1.1.** Un degré Turing $\mathbf{d}$ est *minimal* s'il est différent de $\mathbf{0}$ — le degré calculable — et s'il n'existe pas de degré $\mathbf{e}$ tel que $\mathbf{0} < \mathbf{e} < \mathbf{d}$. ♢

Autrement dit, un ensemble $X \in 2^{\mathbb{N}}$ est de degré minimal s'il est non calculable et si pour toute fonctionnelle $\Phi$ telle que $\Phi(X, n)\downarrow \in \{0, 1\}$ pour tout $n$, soit l'ensemble $\{n : \Phi(X, n)\downarrow = 1\}$ est calculable, soit il permet de calculer $X$.

La preuve de l'existence d'un degré minimal est une des premières utilisations du forcing en calculabilité, via le forcing de Sacks, que nous avons vu dans la section 11-3.

### 1.1. Existence

L'existence de degrés minimaux, due à Spector [215], fut au départ faite par forcing avec des f-arbres $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ calculables *uniformes*, c'est-à-dire des f-arbres tels que pour toute taille $n$ il existe une unique chaîne $\tau_n$ telle que, pour toute chaîne $\sigma$ de taille $n$ et tout $i \in \{0, 1\}$, on a

$$T(\sigma i) = T(\sigma) i \tau_n.$$

On peut aussi voir les chemins de ces f-arbres comme étant toutes les possibilités de complétion d'un ensemble $X$ sur lequel une infinité de bits ne sont pas spécifiés.

La restriction de Spector présente son intérêt pour l'étude plus générale de segments initiaux dans les degrés Turing, mais pour ce qui est des degrés minimaux (c'est-à-dire des segments initiaux de taille 2), Shoenfield [198] a remarqué que le forcing de Sacks calculable, plus simple à manipuler, est suffisant.

**Définition 1.2.** Soit $\Gamma$ une fonctionnelle Turing.

(1) Deux chaînes $\sigma, \tau \in 2^{<\mathbb{N}}$ forment une $\Gamma$-*scission* s'il existe un entier $n$ tel que $\Gamma^\sigma(n)\downarrow \neq \Gamma^\tau(n)\downarrow$.

(2) Un f-arbre $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ est $\Gamma$-*scindé* si pour tout $\sigma \in 2^{<\mathbb{N}}$, les deux chaînes $T(\sigma 0)$ et $T(\sigma 1)$ forment une $\Gamma$-scission.

(3) Un f-arbre $T$ est $\Gamma$-*uni* si aucune paire de chaînes $\sigma, \tau \in \operatorname{Im} T$ ne forme une $\Gamma$-scission. ♢

On considérera pour cette section les fonctionnelles $\Gamma$ comme étant des fonctions partielles de $2^{\mathbb{N}}$ vers $2^{\mathbb{N}}$. Dans ce contexte particulier, on notera alors $\operatorname{dom}\Gamma$ — le domaine de définition de $\Gamma$ — comme étant la classe

$$\{X \in 2^{\mathbb{N}} : \forall n \ \Gamma(X, n)\downarrow \ \in \{0, 1\}\}.$$

Étant donné $X \in \operatorname{dom}\Gamma$, on écrira $\Gamma(X)$ pour l'ensemble

$$\{n \in \mathbb{N} \ : \ \Gamma(X, n)\downarrow = 1\},$$

et l'on parlera de la totalité de $\Gamma$ vis-à-vis de $2^{\mathbb{N}}$, et non pas vis-à-vis de ses entrées pour un oracle fixé.

Le lemme clef dans la construction d'un degré minimal dit que pour toute fonctionnelle $\Gamma$ et tout $f$-arbre calculable, il existe un sous f-arbre calculable sur lequel $\Gamma$ est partout définie et injective, ou sur lequel $\Gamma$ est une fonction constante, restreinte à son domaine de définition dans le sous f-arbre.

**Lemme 1.3.** Pour tout f-arbre calculable $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ et toute fonctionnelle Turing $\Gamma$, il existe un sous f-arbre calculable $S$ de $T$ qui est soit $\Gamma$-scindé, soit $\Gamma$-uni. ⋆

Preuve. Deux cas se présentent.

Cas 1. Il existe une chaîne $\sigma \in 2^{<\mathbb{N}}$ telle pour tous $\rho, \tau \succeq \sigma$, les chaînes $T(\rho)$ et $T(\tau)$ ne forment pas de $\Gamma$-scission. Soit $S$ le sous-f-arbre calculable donné par $S(\mu) = T(\sigma\mu)$. Alors, $S$ est un sous-f-arbre $\Gamma$-uni de $T$.

Cas 2. Pour toute chaîne $\sigma \in 2^{<\mathbb{N}}$, il existe des extensions $\rho, \tau \succeq \sigma$ telles que $T(\rho)$ et $T(\tau)$ forment une $\Gamma$-scission. On calcule alors $S$ comme suit. On définit $S(\epsilon) = T(\epsilon)$. Supposons que l'on ait calculé $S(\sigma) = T(\mu)$ pour des chaînes $\sigma, \mu \in 2^{<\mathbb{N}}$. On cherche alors des chaînes $\rho_0, \rho_1 \succeq \mu$ telles que $T(\rho_0)$ et $T(\rho_1)$ forment une $\Gamma$-scission. Par hypothèse, la recherche aboutit nécessairement. On définit alors $S(\sigma 0) = T(\rho_0)$ et $S(\sigma 1) = T(\rho_1)$. Notons que comme $T(\rho_0)$ et $T(\rho_1)$ forment une $\Gamma$-scission, en particulier, ces deux chaînes sont incomparables.
Ainsi, $S$ est bien un f-arbre, et $\operatorname{Im} S \subseteq \operatorname{Im} T$. Par construction, $S$ est $\Gamma$-scindé. ∎

L'intérêt d'obtenir des f-arbres $\Gamma$-unis ou $\Gamma$-scindés se trouve dans le lemme suivant.

**Lemme 1.4.** Soient $T$ un f-arbre calculable et $\Gamma$ une fonctionnelle.

(1) Si $T$ est $\Gamma$-uni, alors pour tout $X \in \operatorname{dom}\Gamma \cap [T]$ l'élément $\Gamma(X)$ est calculable.

(2) Si $T$ est $\Gamma$-scindé, alors pour tout $X \in \operatorname{dom}\Gamma \cap [T]$ on a $X \leqslant_T \Gamma(X)$. ⋆

Preuve. (1) Supposons que $T$ soit $\Gamma$-uni. Soit $X \in \operatorname{dom}\Gamma \cap [T]$. Alors, pour connaître le $n$-ième bit de $\Gamma(X)$, il suffit de chercher $\sigma \in \operatorname{Im} T$ tel que $\Gamma(\sigma, n) \downarrow = i$ pour $i \in \{0, 1\}$. Le $n$-ième bit de $\Gamma(X)$ est alors égal à $i$.

(2) Supposons que $T$ soit $\Gamma$-scindé, et soit $X \in \operatorname{dom}\Gamma \cap [T]$. Soit $Y = \Gamma(X)$. Pour calculer $X$ à partir de $Y$, on procède comme suit : comme $T(0)$ et $T(1)$ forment une $\Gamma$-scission, il existe $i \in \{0, 1\}$ tel que $\Gamma(T(i), n)$ soit différent de $Y(n)$, pour un certain $n$. On peut trouver $i$ de manière calculable en $Y$. Le bon préfixe de $X$ est alors nécessairement $T(1 - i)$. Supposons que l'on ait calculé un préfixe $\sigma \prec X$ et une chaîne $\tau$ telle que $\sigma = T(\tau)$. Comme $T(\tau 0)$ et $T(\tau 1)$ forment une $\Gamma$-scission, il existe $i \in \{0, 1\}$ tel que $\Gamma(T(\tau i), n)$ soit différent de $Y(n)$, pour un certain $n$. On peut trouver $i$ de manière calculable en $Y$. Le bon préfixe de $X$ est alors nécessairement $T(\tau(1 - i))$. En procédant de la sorte, on calcule alors des préfixes de $X$ de plus en plus grands à partir de $Y = \Gamma(X)$. ∎

Nous avons à présent tous les ingrédients nécessaires pour montrer l'existence de degrés minimaux.

**Théorème 1.5 (Spector [215])**
*Tout ensemble suffisamment générique pour le forcing de Sacks est de degré minimal.*

Preuve. Soit $(\mathbb{P}, \leqslant)$ le forcing de Sacks calculable. Notons d'abord que, d'après l'exercice 11-3.3, si $G \in 2^{\mathbb{N}}$ est suffisamment générique pour ce forcing, alors il n'est pas calculable.

Soit $\Gamma$ une fonctionnelle Turing. D'après le lemme 1.3, l'ensemble des conditions $c \in \mathbb{P}$ telles que $c$ est $\Gamma$-scindé ou $c$ est $\Gamma$-uni est dense. D'après le lemme 1.4, dans le premier cas, pour tout $G \in [c]$, la fonctionnelle $\Gamma$ est définie sur $G$ et $\Gamma(G) \geqslant_T G$. Dans le second cas, d'après le lemme 1.4, pour tout $G \in [c]$, si la fonctionnelle $\Gamma$ est définie sur $G$, alors $\Gamma(G)$ est calculable.

Donc, si $G$ est suffisamment générique pour le forcing de Sacks, il est de degré minimal. ∎

**Corollaire 1.6**
*Il existe une classe parfaite d'ensembles, tous dans des degrés minimaux distincts.*

Preuve. On peut d'abord montrer qu'il existe un arbre parfait de degrés minimaux. Il suffit de procéder comme dans la preuve du théorème 8-5.1, afin de « dupliquer » la construction d'un degré minimal. Avec le formalisme du forcing, soit $(D_n)_{n\in\mathbb{N}}$ une suite d'ensembles denses de conditions du forcing de Sacks, suffisante pour forcer un ensemble à être de degré minimal. On choisit $c_\epsilon \in D_0$, puis pour toute chaîne $\sigma$ de taille $n$, en supposant $c_\sigma \in D_n$ défini, on définit $c_{\sigma 0} \leqslant c_\sigma$ et $c_{\sigma 1} \leqslant c_\sigma$ de telle manière à ce que $[c_{\sigma 0}] \cap [c_{\sigma 1}] = \emptyset$. On pourra également faire en sorte que le premier nœud branchant de $c_{\sigma i}$ étende strictement le premier nœud branchant de $c_\sigma$ pour $i \in \{0, 1\}$. L'arbre final est donné par l'ensemble des chaînes $\tau$ telles qu'il existe $X \in 2^{\mathbb{N}}$ pour lequel $\tau \in \bigcap_{\sigma \prec X} c_\sigma$.

Une fois que l'on a un arbre parfait ne contenant que des degrés minimaux, on peut se reporter à l'exercice 8-5.4 pour en extraire un sous-arbre parfait ne contenant que des degrés deux à deux incomparables. ∎

### 1.2. Calcul d'un degré minimal

Une analyse fine du niveau d'effectivité nécessaire pour mener à bien le théorème 1.5 montre qu'il existe un degré minimal $\emptyset''$-calculable. Il est en fait possible d'améliorer considérablement ce résultat, en remarquant que l'utilisation de f-arbres calculables n'est pas absolument nécessaire.

**Arbre c. e.**

Un $f$-arbre c. e. est une fonction partielle $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ telle que pour tout $\sigma$ tel que $T(\sigma)\downarrow$, soit $T(\sigma 0)\uparrow$ et $T(\sigma 1)\uparrow$, soit $T(\sigma 0)\downarrow \succeq T(\sigma)$ et $T(\sigma 1)\downarrow \succeq T(\sigma)$ avec $T(\sigma 0)$ et $T(\sigma 1)$ incomparables.

L'utilisation de $f$-arbres c. e. $\Gamma$-scindés ou $\Gamma$-unis permet de faire des constructions de degrés minimaux demandant moins de puissance de calcul. Il s'agit alors de constructions effectives qui ne relèvent plus à proprement parler du forcing. Nous listons sans les démontrer trois résultats importants qui utilisent ce nouveau type d'arbre pour calculer plus facilement des ensembles de degrés minimaux. Par « plus facilement », il faut comprendre « avec peu de puissance de calcul », les preuves étant elles au contraire, plus complexes...

**Théorème 1.7 (Sacks [185])**
*Il existe un degré minimal sous $\emptyset'$.*

Notons que, d'après la proposition 7-4.7, le résultat de Sacks implique l'existence d'un degré minimal de degré hyperimmune. Ce résultat a par la suite été amélioré par Yates [235], et indépendamment par Cooper [40].

**Théorème 1.8 (Yates [235], Cooper [40])**
*Soit $X$ un ensemble c. e. non calculable. Alors, $X$ calcule un ensemble $G$ de degré minimal.*

Notons qu'un degré minimal ne peut en revanche jamais être c. e. : il s'agit d'une utilisation sophistiquée de la fameuse méthode des priorités, que nous avons vue dans le chapitre 13, et qui permet de montrer que tout ensemble c. e. non calculable $A$ calcule un autre ensemble c. e. non calculable $B$ qui ne le calcule pas $A$ (voir le théorème 5.1 pour le résultat dans toute sa généralité).

Pour finir, signalons que Groszek et Slaman ont montré que tout degré PA pouvait calculer un degré minimal, via le remarquable résultat suivant.

**Théorème 1.9 (Groszek et Slaman [78])**
*Tout degré PA calcul un degré minimal. Plus précisément, il existe une classe $\Pi^0_1$ non vide dont tous les membres sont soit de degré minimal, soit calculent un degré c. e. non calculable.*

**Exercice 1.10. ($\star$)** Montrer que toute classe $\Pi^0_1$ non vide contient un ensemble de même degré qu'un ensemble c. e. En déduire qu'il n'existe aucune classe $\Pi^0_1$ non vide ne contenant que des éléments de degrés minimaux. $\diamond$

La possibilité de construire des degrés minimaux « complexes » a également été très étudiée. Le principal résultat dans cette direction est le suivant.

**Théorème 1.11 (Kumabe [125])**
*Il existe un degré minimal qui est aussi DNC.*

Kumabe a d'abord montré l'existence d'un degré minimal et DNC, dans un article très complexe, qui ne fut jamais publié. La preuve fut ensuite retravaillée par Kumabe et Lewis [125], et la présentation en a par la suite été simplifiée par Khan et Miller [111], qui l'ont récrite sous le formalisme du forcing, via le *forcing avec arbres touffus*. L'exercice suivant montre qu'un degré minimal ne peut en revanche jamais être $\mathrm{DNC}_2$.

**Exercice 1.12.** $(\star)$ Montrer qu'il existe une classe $\Pi^0_1$ non vide d'ensembles $X \oplus Y$ tels que $X$ est de degré PA et $Y$ et de degré PA relativement à $X$. En déduire qu'aucun ensemble PA n'est de degré minimal (le lecteur pourra consulter la proposition 24-2.4 pour une itération de ce résultat). $\diamond$

### 1.3. Relativisation : couverture minimale

La construction d'un degré minimal avec le forcing sur les f-arbres se relativise à tout degré Turing dans le sens suivant.

**Définition 1.13.** Soient $\mathbf{a}$ et $\mathbf{b}$ des degrés Turing. On dit que $\mathbf{b}$ est une *couverture minimale* de $\mathbf{a}$ si $\mathbf{b} > \mathbf{a}$ et s'il n'existe pas de degré $\mathbf{c}$ tel que $\mathbf{a} < \mathbf{c} < \mathbf{b}$. $\diamondsuit$

Autrement dit, $\mathbf{b}$ est une couverture minimale de $\mathbf{a}$ si c'est un élément minimal dans le cône des degrés Turing strictement au-dessus de $\mathbf{a}$. La relativisation du théorème 1.5 montre le théorème suivant.

**Théorème 1.14**
*Tout degré Turing a une couverture minimale.*

Preuve. Soit $A \subseteq \mathbb{N}$ un ensemble quelconque. L'objectif est de construire un ensemble $B >_T A$ tel que tout ensemble $C$ calculable par $B$ est soit $A$-calculable, soit tel que $A \oplus C \geqslant_T B$. Ainsi, si $B \geqslant_T C >_T A$, on aura bien $C \geqslant_T B$.

Il suffit de considérer une variante du forcing de Sacks, pour lequel nos f-arbres sont cette fois-ci $A$-calculables, et tels que chacun de leurs chemins calcule $A$. On pourra se restreindre par exemple aux f-arbres $A$-calculables tels que $A$ est encodé dans les bits pairs de chaque chemin du f-arbre.

Il suffit alors de répéter la preuve du théorème 1.5 avec ce nouvel ordre partiel, en remarquant la différence suivante : étant donné $T$ un f-arbre $\Gamma$-scindé, pour tout $X \in [T]$, on a à présent besoin de $A$ pour retrouver $X$ à partir de $\Gamma(X)$ : il nous faut en effet la connaissance de $T$. C'est la raison pour laquelle on aura $A \oplus \Gamma(X) \geqslant_T X$ et pas $\Gamma(X) \geqslant_T X$. ∎

Notons qu'une couverture minimale $\mathbf{b}$ de $\mathbf{a}$ n'exclut pas l'existence de degrés $\mathbf{c} < \mathbf{b}$ incomparables avec $\mathbf{a}$. Cela nous amène à définir une notion de couverture plus forte.

**Définition 1.15.** Soient $\mathbf{a}$ et $\mathbf{b}$ deux degrés Turing. On dit que $\mathbf{b}$ est une *couverture minimale forte* de $\mathbf{a}$ si $\mathbf{b} > \mathbf{a}$ et, si pour tout degré Turing $\mathbf{c} < \mathbf{b}$, on a $\mathbf{c} \leqslant \mathbf{a}$. ♢

Comme déjà noté dans la preuve du théorème 1.14, la version relativisée du théorème 1.5 ne prouve pas l'existence d'une couverture minimale forte pour tout degré Turing, et pour cause : certains degrés n'admettent pas de couverture minimale forte, même si ce sera le cas pour beaucoup d'entre eux.

Ishmukhametov [95] a établi une élégante caractérisation des degrés c. e. admettant une couverture minimale forte.

**Théorème 1.16 (Ishmukhametov [95])**
*Un ensemble c. e. $A \subseteq \mathbb{N}$ admet une couverture minimale forte si, et seulement si, toute fonction $A$-calculable $f : \mathbb{N} \to \mathbb{N}$ est bornée pour $n$ suffisamment grand par la fonction $n \mapsto \min\{s \in \mathbb{N} \ : \ \emptyset'[s]\restriction_n = \emptyset'\restriction_n\}$.*

Une caractérisation générale des degrés admettant une couverture minimale forte est pour le moment inconnue, même si de nombreux résultats partiels ont été établis (voir Lewis [143]).

## 2. Nature de $\boldsymbol{\mathcal{D}}$

À quoi ressemble l'ordre partiel $(\boldsymbol{\mathcal{D}}, \leqslant)$ ? Concernant sa taille d'abord, nous avons vu dans le présent ouvrage plusieurs constructions d'arbres parfaits dont chaque chemin est dans un degré Turing différent (voir par exemple l'exercice 7-5.8, l'exercice 8-5.3 ou l'exercice 8-5.4). Cela nous donne une injection de $2^{\mathbb{N}}$ dans $\boldsymbol{\mathcal{D}}$. À l'aide de l'axiome du choix, on peut choisir un représentant dans chaque degré Turing, ce qui donne une injection de $\boldsymbol{\mathcal{D}}$ dans $2^{\mathbb{N}}$. La cardinalité de $\boldsymbol{\mathcal{D}}$ est donc $|2^{\mathbb{N}}|$, celle de $2^{\mathbb{N}}$. Notons que l'on ne peut pas nécessairement choisir un représentant dans chaque degré Turing

si l'on ne dispose pas de l'axiome du choix, et il n'en reste pas moins que « moralement » $|2^{\mathbb{N}}|$ est la cardinalité de $\boldsymbol{\mathcal{D}}$.

Étant donné un élément $\mathbf{a} \in \boldsymbol{\mathcal{D}}$, l'ensemble des éléments sous $\mathbf{a}$ est au plus dénombrable puisqu'un ensemble ne peut calculer qu'une quantité dénombrable d'éléments. La cardinalité des éléments au-dessus de $\mathbf{a}$ est en revanche celle de $2^{\mathbb{N}}$ : étant donné $A \in \mathbf{a}$, on peut facilement créer un arbre parfait dont les chemins sont tous de la forme $A \oplus X$ pour $X \in 2^{\mathbb{N}}$, et tous dans des degrés Turing différents.

L'utilisation de la jointure Turing nous amène à la considération suivante : étant donné deux ensembles $A$ et $B$, l'ensemble $A \oplus B$ calcule à la fois $A$ et $B$, et tout ensemble calculant à la fois $A$ et $B$ calcule $A \oplus B$. En termes de degrés, cela implique que toute paire de degrés $\mathbf{a}, \mathbf{b}$ possède une borne supérieure. Il y a donc un minimum de structure dans cet ordre partiel, pour lequel nous introduisons le concept suivant.

**Définition 2.1.** Un *treillis* est un ensemble partiellement ordonné dans lequel toute paire d'éléments $a, b$ admet une borne supérieure, notée $a \cup b$, et une borne inférieure, notée $a \cap b$. Un *demi-treillis supérieur* (resp. inférieur) est un ensemble ordonné pour lequel toute paire d'éléments admet une borne supérieure (resp. inférieure). ♢

Le paragraphe qui précède cette définition conduit au théorème ci-après, lequel figure dans l'article fondateur de l'étude de la structure des degrés Turing.

**Théorème 2.2 (Kleene et Post [118])**
*L'ordre partiel $(\boldsymbol{\mathcal{D}}, \leqslant)$ est un demi-treillis supérieur de cardinalité $|2^{\mathbb{N}}|$, avec un plus petit mais pas de plus grand élément, tel que chaque élément admet un ensemble au plus dénombrable d'éléments en dessous de lui et un ensemble de cardinalité $|2^{\mathbb{N}}|$ d'éléments au-dessus de lui.*

On calque parfois le vocabulaire des degrés Turing sur celui des ensembles qu'ils contiennent : étant donné deux degrés $\mathbf{a}$ et $\mathbf{b}$, on dira que la borne supérieure $\mathbf{a} \cup \mathbf{b}$ de $\mathbf{a}$ et $\mathbf{b}$ est *la jointure* de $\mathbf{a}, \mathbf{b}$. Notons que dans un demi-treillis supérieur, toute suite finie d'éléments admet également une borne supérieure. En particulier, pour un ensemble fini de degrés $\mathbf{a}_1, \ldots, \mathbf{a}_n$, on la notera $\mathbf{a}_1 \cup \cdots \cup \mathbf{a}_n$, et elle sera le degré de la jointure $A_1 \oplus \cdots \oplus A_n$, pour des représentants quelconques $A_i \in \mathbf{a}_i$.

Que se passe-t-il pour les ensembles dénombrables de degrés ? Le théorème suivant implique que si un tel ensemble est clos par jointure — c'est-à-dire que $\mathbf{a} \cup \mathbf{b}$ est dans notre ensemble pour tous $\mathbf{a}, \mathbf{b}$ dans notre ensemble — et n'a pas d'élément maximal, alors il n'a jamais de borne supérieure.

**Théorème 2.3 (Sacks [188])**
*Tout ensemble dénombrable de degrés clos par jointure et sans élément maximal admet des majorants minimaux, en quantité $|2^{\mathbb{N}}|$.*

Idée de la preuve. Notons d'abord qu'un majorant d'un ensemble dénombrable de degrés $(\mathbf{a}_n)_{n\in\mathbb{N}}$, est aussi majorant de

$$\mathbf{a}_0 \leqslant \mathbf{a}_0 \cup \mathbf{a}_1 \leqslant \mathbf{a}_0 \cup \mathbf{a}_1 \cup \mathbf{a}_2 \leqslant \dots$$

Comme $(\mathbf{a}_n)_{n\in\mathbb{N}}$ n'a pas d'élément maximal et qu'il est clos par jointure, on peut donc considérer sans perte de généralité que notre ensemble de degrés est tel que $\mathbf{a}_n < \mathbf{a}_{n+1}$. Pour tout $n$, soit $A_n$ un représentant de $\mathbf{a}_n$.

Il suffit à présent d'élaborer sur le forcing de Sacks (voir la section 1) permettant de créer la couverture minimale d'un degré. On commence avec un f-arbre $A_0$-calculable dont tous les chemins calculent $A_0$. Une condition de forcing sera un f-arbre $A_n$-calculable dont tous les chemins calculent $A_n$ (pour un certain $n$). On étend une telle condition de forcing $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ au f-arbre $Q$ tel que $\operatorname{Im} Q \subseteq \operatorname{Im} T$ consiste en les chemins qui encodent $A_{n+1} \oplus X$ pour tout $X \in 2^{\mathbb{N}}$. Formellement, pour toute chaîne $\sigma$ et tout $i \in \{0,1\}$, $Q(\sigma i) = T(A_{n+1}\restriction_{|\sigma i|} \oplus \sigma i)$. Comme $A_{n+1} \geqslant_T A_n$, alors $A_{n+1}$ peut calculer $T$, et donc retrouver $A_{n+1} \oplus X$ à partir du chemin de $Q$ qui encode $A_{n+1} \oplus X$ dans $T$.

La manière de faire une couverture minimale ne change pas, et la technique décrite dans la section 1 s'applique de la même manière. L'ensemble générique $G$ obtenu calculera chaque ensemble $A_n$, et sera tel que tout ce qui est calculé par $G$ et qui peut calculer chaque ensemble $A_n$ peut aussi calculer $G$.

Pour obtenir des majorants minimaux en quantité $|2^{\mathbb{N}}|$, on peut construire un arbre parfait de majorants minimaux en subdivisant la construction en deux, puis chaque sous-construction en deux, etc., comme dans la preuve du théorème 8-5.1. ∎

Un ensemble dénombrable de degrés clos par jointure et sans élément maximal a donc toujours deux majorants minimaux distincts, et l'on peut donc en déduire le corollaire suivant.

**Corollaire 2.4**
*Un ensemble dénombrable de degrés clos par jointure et sans élément maximal n'a jamais de borne supérieure.*

Notons que pour une suite d'ensembles $(A_n)_{n\in\mathbb{N}}$, l'ensemble $\bigoplus_{n\in\mathbb{N}} A_n$ (voir la définition 10-3.24) n'est en général par un majorant minimal, comme en témoigne l'élégant résultat suivant.

**Théorème 2.5 (Enderton et Putnam [56], Sacks [190])**
*Il existe un majorant minimal de $(\emptyset^{(n)})_{n\in\mathbb{N}}$ dont le double saut est dans le même degré Turing que celui de $\bigoplus_n \emptyset^{(n)}$.*

Idée de la preuve. Pour une direction, il suffit de remarquer que le double saut de tout majorant de $(\emptyset^{(n)})_{n\in\mathbb{N}}$ permet de calculer $\bigoplus_n \emptyset^{(n)}$. Soit $B$ un majorant et soit $f$ une fonction calculable telle que $f(X') = X$ pour toute $X$ (voir l'exercice 4-6.4 pour plus de détails sur une telle fonction). À l'aide du double saut du majorant $B$, on cherche un code de fonctionnelle $e_1$ tel que $f(\Phi_{e_1}(B)) = \emptyset$, puis un code de fonctionnelle $e_2$ tel que $f(\Phi_{e_2}(B)) = \Phi_{e_1}(B)$, etc.

Pour l'autre direction, il suffit de voir que la construction d'une borne supérieure de $(\emptyset^{(n)})_{n\in\mathbb{N}}$ est effective à l'aide de $\bigoplus_n \emptyset^{(n)}$, et force à chaque étape une fonctionnelle $\Phi_e$ à être partielle ou bien totale sur tous les éléments de l'arbre considéré : on ne fait donc pas que calculer le générique résultant $G$, mais on peut aussi déterminer l'ensemble des codes de fonctionnelles totales sur $G$. Le double saut de $G$ se réduit à cet ensemble (voir l'exercice 5-7.2). ∎

Nous répondons pour finir cette section à la question qui taraude peut-être le lecteur depuis le début de ce chapitre : la structure $(\boldsymbol{\mathcal{D}}, \leqslant)$ est-elle un treillis ? Nous allons voir que non, et nous introduisons pour cela la notion de paire exacte.

**Définition 2.6.** Les degrés $\mathbf{a}, \mathbf{b}$ forment une *paire exacte* pour un ensemble de degrés $C \subseteq \boldsymbol{\mathcal{D}}$ si $\mathbf{a}$ et $\mathbf{b}$ bornent chacun tous les degrés de $C$, et si chaque degré à la fois sous $\mathbf{a}$ et $\mathbf{b}$ est aussi borné par un degré de $C$.♢

**Théorème 2.7**
*Tout ensemble dénombrable de degrés $C$ clos par jointure admet une paire exacte.*

Preuve. Soit $(\mathbf{a}_n)_{n\in\mathbb{N}}$ un ensemble de degrés Turing clos par jointure. Pour tout $n$, soit $A_n$ un représentant de $\mathbf{a}_n$. L'idée est de construire deux ensembles $G_0 = \bigoplus_n X_n^0$ et $G_1 = \bigoplus_n X_n^1$ tels que chaque colonne $X_n^i$ pour $i \in \{0,1\}$ est égale à l'ensemble $A_n$, sauf pour un nombre fini de bits. Il est clair que de tels ensembles $G_0, G_1$ permettent de calculer tous les $A_n$. Il faut à présent les construire via un forcing adapté de telle manière à ce que si $G_0$ et $G_1$ calculent le même ensemble, alors cet ensemble est calculable par $A_0 \oplus A_1 \oplus \cdots \oplus A_m$ pour un certain $m$.
Nos conditions de forcing sont constituées de triplets

$$(\sigma_0, \sigma_1, n), \quad \text{où} \quad \sigma_0, \sigma_1 \in 2^{<\mathbb{N}} \text{ et } n \in \mathbb{N}.$$

Le paramètre $n$ sert à contrôler les extensions possibles de nos conditions. On a $(\sigma_0, \sigma_1, n) \succeq (\tau_0, \tau_1, m)$ pour deux conditions de forcing si $\sigma_0 \preceq \tau_0$, si $\sigma_1 \preceq \tau_1$, et si $n \leqslant m$ avec la restriction que $\forall \langle k, a\rangle$ tel que $|\sigma_i| \leqslant \langle k, a\rangle < |\tau_i|$ pour $k \leqslant n$, on doit avoir $\tau_i(\langle k, a\rangle) = A_k(a)$. Étant donné un ensemble de conditions $(\sigma_0^0, \sigma_1^0, n_0) \succeq (\sigma_0^1, \sigma_1^1, n_1) \succeq (\sigma_0^2, \sigma_1^2, n_2) \succeq \ldots$, le générique $G_0$ sera le point limite de $\sigma_0^0 \preceq \sigma_0^1 \preceq \sigma_0^2 \preceq \ldots$ et de générique $G_1$ sera le point limite de $\sigma_1^0 \preceq \sigma_1^1 \preceq \sigma_1^2 \preceq \ldots$ Notons que la restriction sur les extensions possibles garantit que tant que la suite $n_0 \leqslant n_1 \leqslant n_2 \leqslant \ldots$ est non bornée, chaque $n$-ième colonne de $G_i$ sera bien égale à $A_n$, sauf pour un nombre fini de bits.

Pour toute paire de fonctionnelles $\Phi_{e_0}, \Phi_{e_1}$, on va forcer

$$\Phi_{e_0}(G_0) = \Phi_{e_1}(G_1) = X \implies X \leqslant_T A_0 \oplus \cdots \oplus A_n, \text{ pour un certain } n.$$

Notons que si $G_0$ et $G_1$ bornent le même degré il existe nécessairement deux fonctionnelles telles que $G_0$ et $G_1$ calculent le même ensemble dans ce degré. Une telle construction atteint donc bien nos objectifs.

On se donne une condition de forcing $(\sigma_0, \sigma_1, n)$ et deux fonctionnelles $\Phi_{e_0}$ et $\Phi_{e_1}$. On cherche deux chaînes $\tau_0, \tau_1$ telles que $(\tau_0, \tau_1, n)$ est une extension valide de $(\sigma_0, \sigma_1, n)$, et telles que $\Phi_{e_0}(\tau_0, x) \downarrow \neq \Phi_{e_1}(\tau_1, x) \downarrow$ pour un certain $x$. Si cette recherche aboutit, on prend alors $(\tau_0, \tau_1, n)$ comme extension de $(\sigma_0, \sigma_1, n)$. Sinon, cela signifie que sur tout $x$, les calculs $\Phi_{e_0}(\tau_0, x)$ et $\Phi_{e_1}(\tau_1, x)$, quand ils s'arrêtent, renvoient la même valeur pour toute extension valide $(\tau_0, \tau_1, n) \preceq (\sigma_0, \sigma_1, n)$.

Notons que par définition de ce qu'est une extension valide de $(\sigma_0, \sigma_1, n)$, il est possible de les énumérer à l'aide de $A_0 \oplus \cdots \oplus A_n$. Si pour tout $x$ il existe une extension valide $(\tau_0, \tau_1, n)$ telle que $\Phi_{e_0}(\tau_0, x) \downarrow$ ou $\Phi_{e_1}(\tau_1, x) \downarrow$ alors on peut calculer via $A_0 \oplus \cdots \oplus A_n$ l'unique élément $Z$ ainsi potentiellement calculable par n'importe quel générique $G_0$ via $\Phi_{e_0}$ ou par n'importe quel générique $G_1$ via $\Phi_{e_1}$. Sinon, au moins un des deux calculs $\Phi_{e_0}(G_0, x)$ ou $\Phi_{e_1}(G_1, x)$ sera partiel sur un certain $x$.

Si donc $G_0$, $G_1$ sont suffisamment génériques pour ce forcing, pour toutes fonctionnelles $\Phi_{e_0}$, $\Phi_{e_1}$, on aura

$$\Phi_{e_0}(G_0) = \Phi_{e_1}(G_1) = X \implies X \leqslant_T A_0 \oplus \cdots \oplus A_n, \text{ pour un certain } n.$$

Il s'ensuit que $G_0, G_1$ est une paire exacte pour les degrés $(\mathbf{a}_n)_{n\in\mathbb{N}}$. ∎

On peut à présent en déduire que $(\boldsymbol{\mathcal{D}}, \leqslant)$ n'est pas un treillis.

**Théorème 2.8 (Kleene et Post [118])**
*Le demi-treillis supérieur $\boldsymbol{\mathcal{D}}$ n'est pas un treillis : il y a des degrés $\mathbf{a}, \mathbf{b}$ qui n'ont pas de borne inférieure.*

PREUVE. Il suffit de considérer un ensemble de degrés $\mathbf{c}_0 < \mathbf{c}_1 < \mathbf{c}_2 < \dots$ nécessairement clos par jointure. Cet ensemble de degrés admet donc une paire exacte $\mathbf{a}, \mathbf{b}$. Une telle paire exacte n'a pas de borne inférieure puisque tout degré à la fois sous $\mathbf{a}$ et $\mathbf{b}$ est également sous un degré $\mathbf{c}_n$, pour un certain $n$. ∎

## 3. Universalité de $\mathcal{D}$

Nous voyons dans cette section que $(\boldsymbol{\mathcal{D}}, \leqslant)$ présente une certaine universalité, en ce sens que *tous les ordres partiels* peuvent *se plonger* dans $\boldsymbol{\mathcal{D}}$, sauf ceux qui ne peuvent y prétendre pour des raisons de cardinalité.

**Définition 3.1.** Un ordre partiel $(A, \leqslant)$ se *plonge* dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ s'il existe une injection $f : A \to \boldsymbol{\mathcal{D}}$ telle que $a \leqslant b$ ssi $f(a) \leqslant f(b)$. ♢

La structure $(\boldsymbol{\mathcal{D}}, \leqslant)$ contient donc en elle tous les ordres partiels qui ne sont pas plus gros qu'elle. Cette affirmation, qui sera rendue précise, est en fait un peu fausse : il reste une question ouverte à ce sujet, que nous mentionnerons bientôt. Notons que cela ne nous renseigne pas nécessairement sur la complexité calculatoire de $\boldsymbol{\mathcal{D}}$. Considérons par exemple l'ordre calculable partiel $\leqslant_R \subseteq \mathbb{Q}^2 \times \mathbb{Q}^2$ défini par $(p_1, p_2) \leqslant_R (q_1, q_2)$ si $p_1 \leqslant q_1$ et $p_2 \leqslant q_2$ pour $p_1, p_2, q_1, q_1 \in \mathbb{Q}$. Il n'est pas très difficile de montrer que tout ordre partiel dénombrable se plonge dans $(\mathbb{Q}^2, \leqslant_R)$. La construction d'un plongement se fait sans difficulté, en construisant l'injection de manière gloutonne élément par élément sans jamais violer à chaque étape finie les contraintes d'un plongement. La structure $(\mathbb{Q}^2, \leqslant_R)$ n'est pas calculatoirement complexe, mais elle est suffisamment riche en termes de possibilités pour contenir tous les ordres partiels.

Nous utiliserons à plusieurs reprises l'existence d'ensembles dit *calculatoirement indépendants*.

**Définition 3.2.** Des ensembles $(X_n)_{n\in\mathbb{N}}$ sont *calculatoirement indépendants* si $X_i \not\leqslant_T \bigoplus_{j\neq i} X_j$ pour tout $i \in \mathbb{N}$. ♢

L'existence d'ensembles calculatoirement indépendants ne présente pas de difficultés particulières et l'on peut se référer à l'exercice 10-3.25 pour voir que si $G = \bigoplus_n G_n$ est un ensemble 1-générique, alors les ensembles $(G_n)_{n\in\mathbb{N}}$ sont calculatoirement indépendants.

### 3.1. Plongements dans $\mathcal{D}$

Voyons tout de suite ce qui fut annoncé, sous la forme d'un premier théorème.

**Théorème 3.3 (Sacks [188])**
*Tout ordre partiel dénombrable se plonge dans les degrés Turing.*

Preuve. Nous avons vu dans l'introduction de cette section que tout ordre partiel dénombrable peut se plonger dans la structure $(\mathbb{Q}^2, \leqslant_R)$ définie par $(p_1, p_2) \leqslant_R (q_1, q_2)$ si $p_1 \leqslant q_1$ et $p_2 \leqslant q_2$.

Il suffit alors de montrer que $(\mathbb{Q}^2, \leqslant_R)$ se plonge dans $(\boldsymbol{\mathcal{D}}, \leqslant)$. Soit $(X_n)_{n\in\mathbb{N}}$ une suite d'ensembles calculatoirement indépendants, et soit $(a_n)_{n\in\mathbb{N}}$ une énumération calculable des éléments de $\mathbb{Q}^2$. Le plongement $f$ assigne à l'élément $a_n$ le degré Turing de l'ensemble $\bigoplus_{a_m \leqslant_R a_n} X_m$. On vérifie sans peine $a_n \leqslant_R a_m$ ssi $f(a_n) \leqslant f(a_m)$. ∎

Sacks a par la suite cherché à étendre son résultat à de plus gros ordres partiels. Après tout, $(\boldsymbol{\mathcal{D}}, \leqslant)$ admet pour cardinalité $|2^{\mathbb{N}}|$. On ne peut bien sûr pas attendre de tout ordre partiel de cardinalité $|2^{\mathbb{N}}|$ qu'il se plonge dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ : si un élément dans un ordre partiel a une quantité indénombrable de prédécesseurs, il n'y a aucun espoir de construire un plongement de cet ordre vers $\boldsymbol{\mathcal{D}}$ puisque chaque élément de $\boldsymbol{\mathcal{D}}$ n'en a qu'une quantité dénombrable. On doit donc respecter cette restriction, mais y en a-t-il d'autres ? Nous avons besoin ici d'anticiper un peu sur les ordinaux qui seront introduits dans le chapitre 27, et en particulier sur l'ordinal $\omega_1$, le plus petit ordinal infini non dénombrable. Sacks a obtenu les résultats suivants.

**Théorème 3.4 (Sacks [186])**
*N'importe quel ordre partiel avec une des propriétés suivantes peut se plonger dans les degrés Turing :*

1. *l'ordre est de cardinalité $|2^{\mathbb{N}}|$, et chaque élément a une quantité finie de prédécesseurs ;*
2. *l'ordre est de cardinalité $|\omega_1|$, et chaque élément a une quantité au plus dénombrable de prédécesseurs ;*
3. *l'ordre est de cardinalité $|2^{\mathbb{N}}|$, et chaque élément a une quantité au plus dénombrable de prédécesseurs, ainsi qu'une quantité d'au plus $\omega_1$ successeurs.*

En particulier, si l'on fait l'hypothèse du continu, à savoir $|\omega_1| = |2^{\mathbb{N}}|$, le théorème de Sacks est optimal : tout ordre partiel de cardinalité $|2^{\mathbb{N}}|$, et où chaque élément a une quantité au plus dénombrable de prédécesseurs, peut se plonger dans les degrés Turing. Mais, si l'on n'utilise pas l'hypothèse du continu, la question est toujours ouverte.

**Question 3.5.** Peut-on plonger dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ tout ordre partiel de cardinalité $|2^{\mathbb{N}}|$, où chaque élément a une quantité dénombrable de prédécesseurs ?⋆

Si nous ne présentons pas ici la preuve de Sacks, nous en voyons néanmoins deux ingrédients, via la notion de chaîne et d'anti-chaîne.

**Définition 3.6.** Un ensemble de degrés Turing linéairement ordonné est une *chaîne*. Un ensemble de degrés Turing deux à deux incomparables est une *anti-chaîne*. ♢

**Proposition 3.7 (Sacks [188]).**

(1) Chaque chaîne dénombrable peut être étendue dans les degrés Turing. En particulier, toute chaîne maximale est de cardinalité $\omega_1$.

(2) Chaque anti-chaîne de cardinalité inférieure à $|2^{\mathbb{N}}|$ peut être étendue dans les degrés Turing. En particulier, toute anti-chaîne maximale est de cardinalité $|2^{\mathbb{N}}|$. ⋆

Preuve.

(1) Si la chaîne possède un plus grand élément, on peut considérer son saut Turing. Sinon, on peut considérer le degré de la jointure Turing d'un représentant de chacun de ses éléments.

(2) Soit $D$ l'ensemble des degrés minimaux. Par le corollaire 1.6, $D$ est de cardinalité $|2^{\mathbb{N}}|$. Soit $C$ une anti-chaîne de degrés Turing de cardinalité inférieure à $|2^{\mathbb{N}}|$. Chaque élément de $C$ a une quantité au plus dénombrable d'éléments en dessous de lui. Ainsi, la clôture par le bas de $C$ a la même cardinalité que $C$. Il doit donc exister un élément de $\mathbf{d} \in D$ qui n'est calculé par aucun élément de $C$. Comme $\mathbf{d}$ est un degré minimal, il ne peut borner aucun élément de $C$. On en déduit que $C \cup \{\mathbf{d}\}$ est une anti-chaîne. ∎

### 3.2. Extension de plongements de $\mathcal{D}$

La notion de plongement peut être considérée comme faible, en particulier car elle ne dit rien sur les relations qu'entretiennent les degrés dans l'image d'un plongement, avec les degrés qui ne sont pas dans cette image. Une manière de pallier cette faiblesse est de considérer un plongement déjà existant d'une structure $(C, \leqslant)$ vers les degrés Turing, et d'essayer de voir dans quelle mesure ce plongement peut se prolonger à une extension de l'ordre partiel sur $C$. Une telle chose n'est bien entendue pas toujours possible : si des éléments $a_0, a_1, b \in C$, avec $a_0 < b, a_1 < b$ et $a_0, a_1$ incomparables, sont envoyés sur des degrés $\mathbf{a}_0, \mathbf{a}_1$ et $\mathbf{a}_0 \cup \mathbf{a}_1$, alors un tel plongement ne pourra se prolonger à aucun majorant $c < b$ de $a_0, a_1$. On introduit pour cela la notion d'extension consistante.

**Définition 3.8.** Soit $C$ un demi-treillis supérieur et soit $D \supseteq C$. Alors, $D$ est une *extension consistante* de $C$ si :

(1) pour $a, b < d$ avec $a, b \in C$ et $d \in D \setminus C$, on a $a \cup b < d$ ;

(2) aucun élément de $D \setminus C$ n'est sous un élément de $C$.

Notons que, comme $C$ est un demi-treillis supérieur, $a \cup b \in C$ pour tous $a, b \in C$. ♢

**Théorème 3.9 (Kleene et Post [118])**
*Soit $C$ un demi-treillis supérieur fini, et soit $D$ une extension consistante finie de $C$. Alors, tout plongement $f$ de $(C, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ peut être étendu en un plongement de $(D, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$.*

Preuve. Soit $f$ un plongement de $(C, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$. On notera $\mathbf{a}$ l'image de $a \in C$ par $f$. Soit $a \in D \setminus C$ minimal dans $D \setminus C$. Comme $D$ est une extension consistante, alors $C$ peut être partitionné en une liste d'éléments $(b_i)_{i \leqslant n}$ et $(c_i)_{i \leqslant m}$ telle que $b_0 \cup \cdots \cup b_n < a$, telle que $a$ est incomparable avec chaque $c_i$ et telle que $b_0 \cup \cdots \cup b_n$ n'est au-dessus d'aucun $c_i$. Il suffit alors de construire un degré Turing $\mathbf{a}$ tel que $\mathbf{b}_0 \cup \cdots \cup \mathbf{b}_n < \mathbf{a}$ et tel que $\mathbf{a}$ soit incomparable avec chaque $\mathbf{c}_i$.

En utilisant le fait que $\mathbf{b}_0 \cup \cdots \cup \mathbf{b}_n$ n'est au-dessus d'aucun $\mathbf{c}_i$, on construit facilement par extensions finies un degré $\mathbf{d}$ tel que $\mathbf{d} \cup \mathbf{b}_0 \cup \cdots \cup \mathbf{b}_n$ ne soit au-dessus d'aucun $\mathbf{c}_i$ et tel que $\mathbf{b}_0 \cup \cdots \cup \mathbf{b}_n$ ne soit pas au-dessus de $\mathbf{d}$. Le plongement se prolonge alors en envoyant $a$ sur $\mathbf{d} \cup \mathbf{b}_0 \cup \cdots \cup \mathbf{b}_n$. Par minimalité du choix de $a$, l'ensemble $D - \{a\}$ est à présent une extension consistante de la clôture de $C \cup \{a\}$ en demi-treillis supérieur. On peut donc recommencer jusqu'à assignation de chaque élément de $D$. ∎

Nous verrons avec le lemme 4.2 que la réciproque du théorème fonctionne : il est nécessaire d'être une extension consistante pour que tout plongement soit extensible. Le théorème précédent peut être étendu comme suit.

**Théorème 3.10 (Sacks [186])**
*Soit $C$ un demi-treillis supérieur dénombrable, et soit $f$ un plongement de $(C, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$. Soit $D$ une extension consistante et dénombrable de $C$. Alors, $f$ peut être étendu en un plongement de $(D, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$.*

### 3.3. Segments initiaux de $\boldsymbol{\mathcal{D}}$

Une autre manière de renforcer l'étude des plongements possibles, est de considérer les plongements sur des *segments initiaux* de $\boldsymbol{\mathcal{D}}$

**Définition 3.11.** Un *segment initial* de $\boldsymbol{\mathcal{D}}$ est un ensemble de degrés clos par le bas. Un *segment final* est un ensemble de degrés clos par le haut.♢

Un plongement sur un segment initial nous donne une information complète sur tous les degrés qui se trouvent en dessous de ceux de l'image du plongement. Par exemple, la construction d'un degré minimal nous indique que l'ordre $a < b$ de deux éléments $a, b$ peut se plonger sur un segment initial de $\boldsymbol{\mathcal{D}}$. L'existence d'un degré minimal avec une couverture minimale forte (voir la définition 1.15) nous indique que l'ordre $a < b < c$ de trois éléments $a, b, c$ peut se plonger sur un segment initial de $\boldsymbol{\mathcal{D}}$. En élaborant sur la construction des degrés minimaux, Lachlan et Lebeuf ont obtenu le remarquable résultat suivant.

**Théorème 3.12 (Lachlan et Lebeuf [133])**
*Un ordre partiel dénombrable se plonge sur un segment initial de* $(\boldsymbol{\mathcal{D}}, \leqslant)$ *si, et seulement si, c'est un demi-treillis supérieur avec un plus petit élément.*

La preuve de Lachlan et Lebeuf se fait petit à petit, l'étape la plus difficile étant de le montrer pour tout demi-treillis supérieur fini avec un plus petit élément. Il s'agit d'une modification non triviale de la construction de degrés minimaux. Considérons par exemple l'ordre partiel en diamant donné par $a \leqslant b_1$, $a \leqslant b_2$, $b_1, b_2 \leqslant c$ et $b_1, b_2$ incomparables. Il s'agit alors de construire deux degrés minimaux $\mathbf{b}_1, \mathbf{b}_2$, tels que $\mathbf{b}_1 \cup \mathbf{b}_2 = \mathbf{c}$, et tels que $\mathbf{b}_1, \mathbf{b}_2$ sont *les seuls degrés non calculables* se trouvant sous $\mathbf{c}$. Une telle construction repose sur un forcing avec des f-arbres calculables dits *uniformes*, comme expliqué dans la section 1.1. On peut par exemple construire avec un tel forcing un ensemble $X = X_0 \oplus X_1$ tel que $X_0$ et $X_1$ sont incomparables et minimaux, et tel que tout ce qui est calculé par $X$ est soit sous $X_0$, soit sous $X_1$, soit peut recalculer $X$. La preuve détaillée peut être consultée dans [169] ou [138].

Notons que le théorème de Lachlan et Lebeuf donne une caractérisation complète des idéaux Turing de la forme $\boldsymbol{\mathcal{D}}(\leqslant \mathbf{a})$ pour un certain degré $\mathbf{a}$ (i.e. l'ensemble des éléments qui se trouvent sous $\mathbf{a}$) : il s'agit des demi-treillis supérieurs au plus dénombrables ayant un plus petit et un plus grand élément. En fait, ce théorème peut être poussé un peu plus loin.

**Théorème 3.13 (Abraham et Shore [2])**
*Un ordre partiel de cardinalité* $|\omega_1|$ *est isomorphe à un segment initial de* $(\boldsymbol{\mathcal{D}}, \leqslant)$ *ssi c'est un demi-treillis supérieur avec un plus petit élément, et dans lequel chaque élément a une quantité au plus dénombrable de prédécesseurs.*

En particulier, si l'on fait l'hypothèse du continu, cela caractérise complètement les segments initiaux possibles de $(\boldsymbol{\mathcal{D}}, \leqslant)$. Si l'on ne fait pas l'hypothèse du continu, alors les choses se compliquent.

**Théorème 3.14 (Groszek et Slaman [77])**
*Il existe un modèle de ZFC, dans lequel il existe un ordre partiel indénombrable avec un plus petit élément et pour lequel chaque élément a un nombre fini de prédécesseurs, qui ne peut se plonger comme segment initial de* $(\boldsymbol{\mathcal{D}}, \leqslant)$.

## 4. Théorie du premier ordre de $\boldsymbol{\mathcal{D}}$

Nous abordons à présent la complexité de $\boldsymbol{\mathcal{D}}$. La question qui nous intéresse est la suivante : étant donné un énoncé du premier ordre qui porte sur les degrés Turing, peut-on décider si ce dernier est vérifié ou non ? Le langage que nous pouvons utiliser est uniquement constitué des relations $\leqslant, <$ ou $=$, mais il sera possible d'étendre ce langage à tout ce qui est définissable avec $\leqslant, <$ ou $=$. Par exemple $\mathbf{0}$, le degré minimal, est définissable comme étant l'unique degré qui satisfait la formule $F(\mathbf{x}) = \forall \mathbf{y}\ \mathbf{x} \leqslant \mathbf{y}$. On peut alors par exemple exprimer l'existence d'un degré minimal de la manière suivante : $\exists \mathbf{x}\ \big(\mathbf{0} < \mathbf{x} \wedge \forall \mathbf{y} \leqslant \mathbf{x}\ (\mathbf{y} = \mathbf{0} \vee \mathbf{y} = \mathbf{x})\big)$. Le fait que $\mathbf{0}$ soit définissable par la formule $F(\mathbf{x})$ permet de s'en passer dans une formule équivalente :

$$\exists \mathbf{z}\ \Big(F(\mathbf{z}) \wedge \exists \mathbf{x}\ \big(\mathbf{z} < \mathbf{x} \wedge \forall \mathbf{y} \leqslant \mathbf{x}\ (\mathbf{y} = \mathbf{z} \vee \mathbf{y} = \mathbf{x})\big)\Big).$$

C'est bien entendu plus long, et l'on s'autorisera donc ces extensions de langage. On utilisera notamment la fonction à deux arguments $\cup$ qui nous donne la jointure de deux degrés, et qui est elle aussi définissable par la formule

$$F(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x} \leqslant \mathbf{z} \wedge \mathbf{y} \leqslant \mathbf{z} \wedge \forall \mathbf{a}\ \big((\mathbf{x} \leqslant \mathbf{a} \wedge \mathbf{y} \leqslant \mathbf{a}) \to \mathbf{z} \leqslant \mathbf{a}\big).$$

On vérifie sans peine que pour tout $\mathbf{a}, \mathbf{b} \in \boldsymbol{\mathcal{D}}$, le degré $\mathbf{a} \cup \mathbf{b}$ est l'unique degré $\mathbf{z}$ tel que $F(\mathbf{a}, \mathbf{b}, \mathbf{z})$.

Revenons à présent à notre question : étant donné un énoncé sur les degrés Turing, peut-on décider si ce dernier est vrai ou faux ? La question est *a priori* d'une grande complexité si on l'aborde directement : un énoncé du type $\exists \mathbf{a}$ revient à un énoncé du type « existe-t-il un ensemble $X$ tel que ». Il s'agit d'une quantification du second ordre, c'est-à-dire portant non pas sur les entiers, mais sur les ensembles d'entiers. Ce genre de quantification sera abordé en détail dans la partie IV. Les énoncés avec des quantifications sur les entiers étant déjà indécidables, il y a fort à parier que ce soit aussi le cas pour les énoncés sur les degrés Turing. On peut néanmoins montrer que si la formule est du type $\forall\exists$ ou du type $\exists\forall$, on peut alors décider si elle est vrai ou fausse. Cela constitue notre premier théorème sur le sujet.

**Théorème 4.1 (Lerman [138] (théorème 4.4) et Shore [199])**
*La théorie $\Pi^0_2$ de $(\boldsymbol{\mathcal{D}}, \leqslant)$, c'est-à-dire l'ensemble des énoncés $\Pi^0_2$ vrais dans $(\boldsymbol{\mathcal{D}}, \leqslant)$, est décidable.*

Pour montrer le théorème précédent, il suffit essentiellement de voir que la réciproque du théorème 3.9 est vraie.

**Lemme 4.2.** Soit $C$ un demi-treillis supérieur fini, et soit $D$ une extension finie de $C$. Alors, $D$ est une extension consistante si, et seulement si, tout plongement $f$ de $(C, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ peut être étendu en un plongement de $(D, \leqslant)$ dans $(\boldsymbol{\mathcal{D}}, \leqslant)$. ⋆

Preuve. Le théorème 3.9 nous donne une direction du lemme. Supposons à présent que $D$ ne soit pas une extension consistante de $C$. Si un élément de $d \in D$ est tel que $a, b \leqslant d$ mais $a \cup b \not\leqslant d$, il suffit de considérer un plongement qui associe $\mathbf{a} \cup \mathbf{b}$ à $a \cup b$. Il sera alors impossible d'étendre un tel plongement à $D$. Supposons à présent qu'un degré de $D$ se trouve sous un degré de $C$. Par le théorème 3.12, il existe un plongement de $C$ sur un segment initial de $\boldsymbol{\mathcal{D}}$. Là encore, un tel plongement ne pourra pas être étendu à un degré inférieur à un degré de $C$. ∎

Nous avons à présent l'ingrédient nécessaire pour montrer le théorème 4.1 :

Preuve du théorème 4.1. Soit un énoncé de la forme

$$\forall c_1, \ldots, \forall c_n \ \exists d_1, \ldots, d_m \ F(c_1, \ldots, c_n, d_1, \ldots, d_m),$$

où $F$ est une combinaison booléenne de formules atomiques. Soit $C$ l'ensemble fini des modèles possibles de demi-treillis supérieurs générés par les $c_i$ et compatibles avec les conditions données par $F$. Si $C$ est vide, alors la formule n'est pas satisfaite, sans même considérer la partie de $F$ mentionnant les $d_i$. Sinon, pour tous les modèles de $C$, on vérifie si ce modèle peut être complété de manière compatible avec les conditions données par $F$, et de telle manière à ce que les $d_i$ en soient une extension consistante. Si tel est le cas, la formule est vraie, et sinon elle est fausse. ∎

Il s'agit là de la limite de ce qui est décidable. Lachlan [130] a montré que la théorie $\Pi^0_3$ ne l'était plus.

**Théorème 4.3 (Schmerl — voir corollaire 4.6 de [138])**
*La théorie $\Pi^0_3$ de $(\boldsymbol{\mathcal{D}}, \leqslant)$ est indécidable.*

À quel point la théorie de $(\boldsymbol{\mathcal{D}}, \leqslant)$ est-elle complexe ? Peut-on par exemple la décider à l'aide du saut Turing, ou encore à l'aide de la réunion disjointe

de toutes les itérations finies du saut Turing ? Nous allons voir que non : la théorie de $(\boldsymbol{\mathcal{D}}, \leqslant)$ est de complexité *maximale*. Qu'entendons-nous par là ? Considérons $T_2$, la théorie du second-ordre de $(\mathbb{N}, \times, +, 0, 1)$, c'est-à-dire l'ensemble des formules du second ordre qui sont vraies dans $\mathbb{N}$. On rappelle qu'une formule du second ordre est de la forme $\forall X\ \exists Y\ \ldots F(X, Y, \ldots)$ où les variables $X, Y, \ldots$ sont des ensembles d'entiers, et où $F$ est une formule du premier ordre, paramétrée par ces ensembles.

La théorie $T_2$ est donc l'ensemble des formules du second ordre qui sont vraies dans $\mathbb{N}$. Si l'on a accès à $T_2$, on peut savoir si une formule de la théorie du premier ordre de $(\boldsymbol{\mathcal{D}}, \leqslant)$ est vraie : les quantifications $\exists \mathbf{a}$ et $\forall \mathbf{a}$ peuvent se remplacer par des quantifications sur les éléments de $2^{\mathbb{N}}$ et, à l'aide du théorème 9-3.4 relativisé aux paramètres du second ordre — lequel permet de transformer un prédicat $\Sigma^0_n(X, Y, \ldots)$ en une formule $\Sigma_n$ de l'arithmétique —, on peut transformer une formule du premier ordre $F$ de $(\boldsymbol{\mathcal{D}}, \leqslant)$ en une formule de second ordre équivalente $F^*$ de $(\mathbb{N}, \times, +, 0, 1)$. Simpson a montré que l'inverse était vrai aussi : via un codage ingénieux, il est possible de transformer une formule de l'arithmétique du second ordre en une formule équivalente de $(\boldsymbol{\mathcal{D}}, \leqslant)$.

Insistons avant d'aller plus loin sur la complexité *extrême* de $T_2$. Nous étudierons dans la partie IV tous les détails de la complexité de $T_2$ restreinte aux formules $\Pi^1_1$, c'est-à-dire restreinte aux formules au sein desquelles les quantifications du second ordre sont toutes universelles. Nous verrons que cette théorie a déjà un degré Turing considérablement élevé comparé à $\emptyset'$ ou même à toutes les itérations finies de $\emptyset'$. Ce degré Turing est néanmoins bien défini, et il est *absolu* dans le sens où la valeur de vérité d'une formule $\Pi^1_1$ sera la même dans les modèles transitifs de la théorie des ensembles partageant les mêmes ordinaux calculables (voir la partie IV pour une définition formelle). À partir du niveau de complexité $\Pi^1_2$ des formules, ce sens devient plus flou. La valeur de vérité de ce genre de formule restera toutefois inchangée dans tous les modèles transitifs de la théorie des ensembles qui partagent cette fois non pas les même ordinaux calculables, mais les même ordinaux dénombrables. Sous réserve d'accepter l'absoluité des ordinaux dénombrables, la vérité des formules $\Pi^1_2$ est elle aussi absolue. Le degré Turing du niveau $\Pi^1_2$ de $T_2$ est quant à lui plus élevé que tous les degrés Turing abordés dans le présent livre (il s'agit en quelque sorte du supremum de tous les singletons $\Pi^1_1$, dont nous verrons la définition dans la section 30-4). La valeur de vérité d'une formule $\Pi^1_3$ pourra quant à elle différer entre deux modèles de ZFC qui partagent les mêmes ordinaux, et le sens qu'il y a à dire qu'une telle formule est *vraie* ou *fausse* s'évanouit ici encore un peu plus. Quant au degré Turing de la théorie $\Pi^1_3$ de $T_2$, de là où nous sommes, c'est-à-dire le monde des choses calculables, depuis lequel

nous observons la structure de l'univers, même les meilleurs télescopes ne permettent pas de le voir : il est tout simplement trop éloigné de nous.

Voilà donc la complexité de la théorie des degrés Turing ! Nous livrons ci-après la preuve moderne du théorème de Simpson, qui diffère de celle qui fut originellement produite, et qui présente son intérêt propre. Le résultat de Simpson découle du théorème suivant, où $n \in \mathbb{N}$..

**Théorème 4.4 (Slaman et Woodin [206])**
*Tout sous-ensemble dénombrable $R \subseteq \boldsymbol{\mathcal{D}}^n$ de $n$-uplets de degrés Turing est uniformément définissable dans $(\boldsymbol{\mathcal{D}}, \leqslant)$ avec un nombre fini de paramètres. Formellement, il existe une formule $F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ telle que, pour tout $R \subseteq \boldsymbol{\mathcal{D}}^n$, il existe des paramètres $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \boldsymbol{\mathcal{D}}$ pour lesquels $(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \boldsymbol{\mathcal{D}}$ si, et seulement si, $F(\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{p}_1, \ldots, \mathbf{p}_m)$ est vrai dans $(\boldsymbol{\mathcal{D}}, \leqslant)$.*

Voyons tout de suite comment utiliser le théorème 4.4 pour montrer le théorème de Simpson : il suffit de coder un modèle standard de l'arithmétique dans les degrés Turing.

**Théorème 4.5 (Simpson [201])**
*La théorie du premier ordre des degrés Turing est many-one équivalente à celle de l'arithmétique du second ordre.*

Preuve. Nous avons déjà vu dans les paragraphes précédents comment transformer un énoncé de $(\boldsymbol{\mathcal{D}}, \leqslant)$ en un énoncé équivalent de l'arithmétique du second ordre. Voyons à présent comment faire l'inverse.

L'idée est de coder un modèle standard de $(\mathbb{N}, +, \times, 0, 1)$ dans les degrés Turing. Un tel modèle sera codé par un ensemble fini de paramètres codant pour un ensemble $N$ de degrés Turing qui représentent $\mathbb{N}$, avec un degré spécifique représentant 0 et un autre représentant 1. Les relations $+$ et $\times$ sont elles aussi codées par un ensemble fini de paramètres.

Il est possible de créer une formule de $\boldsymbol{\mathcal{D}}$ qui vérifie si un ensemble fini de paramètres code bien pour le modèle standard de l'arithmétique : il s'agit simplement de vérifier les axiomes de l'arithmétique de Robinson (voir la section 9-2.3), lesquels sont en nombre fini, et de vérifier ensuite que tout sous-ensemble du modèle possède un plus petit élément. On peut se reporter au théorème 9-3.13 pour voir que ces conditions sont nécessaires et suffisantes pour vérifier que l'on a bien affaire au modèle standard des entiers. La quantification universelle « tout sous-ensemble du modèle possède un plus petit élément » peut être remplacée par une quantification universelle sur les degrés Turing utilisés comme paramètres pour coder des sous-ensembles de $N$ (notre ensemble de degrés qui représente $\mathbb{N}$).

Étant donné une formule $F$ de l'arithmétique du second ordre, on peut finalement la transformer en une formule équivalente $F^*$ dans $\boldsymbol{\mathcal{D}}$, en remplaçant les quantifications sur les ensembles par des quantifications sur les paramètres codant pour ces ensembles. La formule du second ordre de l'arithmétique sera donc interprétée dans $\boldsymbol{\mathcal{D}}$ par la formule : il existe des paramètres codant pour un modèle standard de l'arithmétique, tel que $F^*$ est vérifiée dans ce modèle. ∎

Passons à présent au codage de Slaman et Woodin. Le lemme qui suit constitue la partie difficile de la preuve. Il repose sur un forcing qui peut sembler relativement simple dans son principe, mais dont l'exécution s'avère délicate et demande pas mal d'astuce pour être menée à bien.

**Lemme 4.6 (Slaman et Woodin [206]).** Toute anti-chaîne dénombrable dans les degrés Turing est uniformément définissable avec trois paramètres. ⋆

Preuve. Soit $(\mathbf{a}_n)_{n\in\mathbb{N}}$ une anti-chaîne dans les degrés Turing, et soit $\mathbf{b}$ un majorant de cette anti-chaîne. Nous allons définir deux degrés $\mathbf{g}_0, \mathbf{g}_1$ tels que pour tout degré $\mathbf{y} \leqslant \mathbf{b}$ ne majorant aucun $\mathbf{a}_i$, alors $\mathbf{g}_0 \cup \mathbf{y}$ et $\mathbf{g}_1 \cup \mathbf{y}$ ont une borne inférieure, et cette borne inférieure est $\mathbf{y}$. En d'autre termes, tout degré à la fois sous $\mathbf{g}_0 \cup \mathbf{y}$ et sous $\mathbf{g}_1 \cup \mathbf{y}$ doit aussi être sous $\mathbf{y}$. À l'inverse, il existera pour tout $i$ un degré à la fois sous $\mathbf{g}_0 \cup \mathbf{a}_i$ et sous $\mathbf{g}_1 \cup \mathbf{a}_i$ qui ne sera pas sous $\mathbf{a}_i$. Il s'ensuit que chaque $\mathbf{a}_i$ sera un élément minimal satisfaisant la formule

$$F(\mathbf{x}) = \mathbf{x} \leqslant \mathbf{b} \ \wedge \exists \mathbf{c}\ (\mathbf{c} \not\leqslant \mathbf{x} \ \wedge \ \mathbf{c} \leqslant \mathbf{g}_0 \cup \mathbf{x} \ \wedge \ \mathbf{c} \leqslant \mathbf{g}_1 \cup \mathbf{x}).$$

En particulier, les degrés $\mathbf{a}_i$ seront exactement les degrés $\mathbf{x}$ satisfaisant la formule $F(\mathbf{x}) \wedge \forall \mathbf{y} \leqslant \mathbf{x}\ \neg F(\mathbf{y})$. Le fait que les $\mathbf{a}_i$ forment une anti-chaîne est utilisé uniquement pour les définir comme solutions minimales de $F$, mais n'intervient plus par la suite.

Nous utiliserons pour la construction des degrés $\mathbf{g}_0$ et $\mathbf{g}_1$ le fait suivant : tout degré Turing contient un ensemble $X$ calculable en n'importe quel sous-ensemble infini de $X$. On peut le voir de la manière suivante : étant donné un ensemble $Y$ quelconque, on définit $X$ comme étant l'ensemble des préfixes $\sigma \prec Y$, via un codage des chaînes finies par des entiers.

Soit $B$ un représentant de $\mathbf{b}$ et, pour tout $n$, soit $A_n$ un représentant de $\mathbf{a}_n$ calculable en n'importe lequel de ses sous-ensembles infinis. Nous allons définir deux ensembles $G_0, G_1$ tels que pour tout $i$ il existe $C \not\leqslant_T A_i$ tel que $C \leqslant_T G_0 \oplus A_i$ et $C \leqslant_T G_1 \oplus A_i$, et tel que pour tout $Y \leqslant_T B$ et tout $D$ tel que $D \leqslant_T G_0 \oplus Y$ et $D \leqslant_T G_1 \oplus Y$, alors $Y \geqslant_T D$ ou bien $Y \geqslant_T A_j$ pour un certain $j$.

On procède à cet effet via un forcing qui présente des similarités avec celui du théorème 2.7.

Soit $\mathbb{P}$ l'ensemble de conditions de la forme $(\sigma_0, \sigma_1, n)$ pour $\sigma_0, \sigma_1 \in 2^{<\mathbb{N}}$, avec $|\sigma_0| = |\sigma_1|$ et $n \in \mathbb{N}$. L'entier $n$ sert à restreindre les extensions possibles, la chaîne $\sigma_0$ est utilisée pour le premier générique $G_0$ et la chaîne $\sigma_1$ pour le deuxième générique $G_1$. Tout comme dans la preuve du théorème 2.7, on peut voir $G_0$ et $G_1$ comme étant construits par colonne. L'entier $n$ indique que la construction sera dorénavant restreinte sur les $n$ premières colonnes : pour une colonne $k \leqslant n$, si $a \notin A_k$, il n'y a alors aucune restriction pour le bit $\langle k, a\rangle$ des deux génériques. Si en revanche $a \in A_k$, alors le bit $\langle k, a\rangle$ des deux génériques doit être identique (sans nécessairement être égal à $A_k(a)$).
Formellement, $(\sigma_0, \sigma_1, n) \succeq (\tau_0, \tau_1, m)$ si $\sigma_0 \preceq \tau_0$, si $\sigma_1 \preceq \tau_1$, si $n \leqslant m$, et si de plus la condition suivante est vérifiée : pour tout $k \leqslant n$, alors pour tout $a \in A_k$ tel que $|\sigma_i| \leqslant \langle k, a\rangle < |\tau_i|$, les valeurs $\tau_0(\langle k, a\rangle)$ et $\tau_1(\langle k, a\rangle)$ doivent être les mêmes.

Considérons $G_0, G_1$ deux ensembles suffisamment génériques pour ce forcing. Pour chaque $A_n$, et pour $a \in A_n$ suffisamment grand, on aura

$$G_0(\langle n, a\rangle) = G_1(\langle n, a\rangle),$$

par définition de ce qu'est une extension valide dans ce forcing. En particulier ; les ensembles $X_0^n, X_1^n \subseteq A_n$ définis par $X_i^n(a) = 0$ si $a \notin A_n$, et $X_i^n(a) = G_i(\langle n, a\rangle)$ sinon, sont les mêmes sauf pour un nombre fini de bits, et sont donc tous les deux calculés par $G_0 \oplus A_n$ et $G_1 \oplus A_n$.

Montrons que si $G_0, G_1$ sont suffisamment génériques, alors aucun $A_n$ ne peut calculer les ensembles $X_0^n, X_1^n$ ainsi définis. Étant donné une condition $\langle \sigma_0, \sigma_1, n\rangle$, on peut prendre n'importe quelle extension pour le côté $\sigma_0$, ce qui force alors certains bits de l'extension pour l'autre côté. En considérant le fait qu'il y a nécessairement des sous-ensembles infinis de $A_n$ non calculables en $A_n$, on peut nécessairement trouver une extension $\tau_0 \succeq \sigma_0$ telle que pour une fonctionnelle $\Phi_e$ donnée, $\Phi_e(A_n)$ ne produise jamais la restriction de $\tau_0$ qui sera faite pour en faire un préfixe de $X_0^n$ — soit parce que $\Phi_e(A_n)$ sera partielle, soit parce qu'elle produira une chaîne incompatible avec le préfixe de $X_0^n$ ainsi forcé. Comme $X_1^n$ coïncide avec $X_0^n$ sauf sur un nombre fini de bits, alors $A_n$ ne calculera pas non plus $X_1^n$. Cela nous donne la première partie de ce que l'on cherche à montrer : pour tout $A_n$, il existe un ensemble calculable en $G_0 \oplus A_n$ et en $G_1 \oplus A_n$, mais pas en $A_n$.

Il reste à montrer que pour tout $Y \leqslant_T B$ tel que $Y$ ne calcule aucun $A_n$, si $G_0 \oplus Y$ et $G_1 \oplus Y$ calculent un même ensemble $C$, alors $Y \geqslant_T C$. Soit alors $p = (\sigma_0, \sigma_1, n)$ une condition et soient $\Phi_{e_0}, \Phi_{e_1}$ une paire de fonctionnelles. On sépare dans un premier temps la chaîne $\sigma_0$ de notre condition $p$. S'il existe $x$ et $\tau_0 \succeq \sigma_0$ tels que pour tout $\rho_0 \succeq \tau_0$ on a $\Phi_{e_0}(Y \oplus \rho_0, x)\uparrow$, on considère alors une chaîne $\tau_1$ telle que la condition $(\tau_0, \tau_1, n)$ forme une extension valide, pour laquelle on aura forcé la partialité de $\Phi_{e_0}(Y \oplus G_0)$.

Supposons à présent que pour tout $x$ et pour tout $\tau_0 \succeq \sigma_0$ il existe $\rho_0 \succeq \tau_0$ tel que $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow$. Supposons dans un premier temps qu'il existe une extension $\tau \succeq \sigma_0$ telle que, pour toutes extensions $\rho_0, \rho_1 \succeq \tau$ et pour tout $x$, on ait $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow = a$ et $\Phi_{e_0}(Y \oplus \rho_1, x) \downarrow = b$ impliquent $a = b$. Alors, on ne peut produire qu'un unique ensemble via $\Phi_{e_0}(Y \oplus G_0)$ pour un générique $G_0$ quelconque, et cet ensemble est calculable alors en $Y$. On force donc $\Phi_{e_0}(Y \oplus G_0)$ à calculer quelque chose qui est déjà calculable en $Y$. Supposons finalement que, pour tout $\tau \succeq \sigma_0$, il existe $\rho_0, \rho_1 \succeq \tau$ et $x$ tels que $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow \neq \Phi_{e_0}(Y \oplus \rho_1, x) \downarrow$. Le lemme suivant s'avère utile.

**Lemme 4.7.** Pour tout $\tau \succeq \sigma$, il existe $x$ et deux extensions $\rho_0, \rho_1 \succeq \tau$ qui diffèrent sur seulement un bit et tels que

$$\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow = a \neq \Phi_{e_0}(Y \oplus \rho_1, x) \downarrow = b. \qquad \star$$

Preuve. Il suffit de trouver deux extensions de $\tau$ de même taille et incompatibles sur un certain $x$. Soient $i_0, \ldots, i_k$ les bits sur lesquelles ces extensions diffèrent. On inverse le bit $i_0$ dans la première extension, et on l'étend pour obtenir une valeur $a_0$ pour $x$. Si $a_0 \neq a$, on a terminé. Sinon, on inverse à son tour $i_1$ dans cette nouvelle extension, que l'on étend encore pour obtenir une valeur $a_1$, et ainsi de suite. Si chaque valeur $a_1 = a_2 = \cdots = a_{k-1}$, alors $a_{k-1} \neq b$, et notre chaîne diffère maintenant d'un seul bit de celle qui a produit $b$. ∎

On se sert du lemme précédent pour calculer à l'aide de $Y$ une suite de quadruplets $(\tau_{0,m}, \tau_{1,m}, i_m, x_m)_{m \in \mathbb{N}}$ avec $i_m < i_{m+1}$ telle que, pour tout $m$, les chaînes $\tau_{0,m}, \tau_{1,m}$ diffèrent sur exactement le bit $i_m$ et soient incompatibles sur $x_n$. La suite de bits $(i_m)_{m \in \mathbb{N}}$ est une suite infinie et $Y$-calculable. Rappelons que notre condition de forcing est de la forme $(\sigma_0, \sigma_1, n)$. S'il existe $i_m$ tel que $i_m = \langle k, a \rangle$ pour $k > m$, cela entraîne que pour toute extension $\tau'$ de $\sigma_1$ telle que $\langle \tau_{0,m}, \tau', n \rangle$ est une extension valide, alors $\langle \tau_{1,m}, \tau', n \rangle$ en est aussi une, car il n'y a aucune contrainte sur le bit $i_m$. On peut donc trouver une extension de $\tau'$ de $\sigma_1$ qui force une valeur pour $x_n$ (à supposer que l'on ne puisse pas forcer la partialité de ce côté là), et l'on prend l'extension $\tau_{0,m}$ ou $\tau_{1,m}$ de $\sigma_0$ qui force une valeur différente. On force donc $\Phi_{e_0}(Y \oplus G_0)$ et $\Phi_{e_1}(Y \oplus G_1)$ à être différents.

Si à présent il n'existe pas $i_m = \langle k, a \rangle$ pour $k > m$, alors il doit exister par le principe des tiroirs un certain $k \leqslant m$ pour lequel une infinité de $i_m$ est de la forme $\langle k, a \rangle$. Aussi n'est-il pas possible d'avoir $A_k(a) = 1$ pour chacun de ces $i_m$, car on aurait alors un sous-ensemble infini de $A_k$ et $Y$-calculable, or par hypothèse tout sous-ensemble infini de $A_k$ calcule $A_k$, et $Y$ ne calcule pas $A_k$. Il doit donc exister $\tau_{0,m}, \tau_{1,m}$ et $i_m = \langle k, a \rangle$ tel que $A_k(a) = 0$. Là encore, toute extension $\tau'$ de $\sigma_1$ qui est compatible avec $\tau_{0,m}$ le sera aussi avec $\tau_{1,m}$, car il n'y a aucune contrainte sur le bit $i_m$. On peut alors trouver

une extension $\tau'$ de $\sigma_1$ qui force une valeur sur $x_m$ —à moins que l'on ne puisse forcer la partialité de ce côté là— et choisir une extension parmi $\tau_{0,m}$ et $\tau_{1,m}$ qui force une autre valeur sur $x_m$. Cela conclut la preuve. ∎

Et voilà. La preuve du lemme ne fut pas sans difficulté, mais nous sommes à présent presque au bout de nos peines. Montrons finalement que tout sous-ensemble dénombrable de $\boldsymbol{\mathcal{D}}^n$ peut être codé dans les degrés Turing.

Preuve du théorème 4.4. Dans ce qui suit, les variables en majuscule dénotent des ensembles ou suites de degrés Turing. Soit un ensemble dénombrable de $n$-uplets $R \subseteq \boldsymbol{\mathcal{D}}^n$. Soit $\mathbf{b}$ un majorant sur l'ensemble des degrés concernés par $R$ (c'est-à-dire sur la réunion des projections de $R$ sur chaque coordonnée), et soit $(\mathbf{x}_i)_{i\in\mathbb{N}}$ une liste de tous les degrés sous $\mathbf{b}$. Notons que $\mathbf{b}$ n'est qu'un majorant, et que certains $\mathbf{x}_i$ peuvent donc ne pas être des degrés du $n$-uplet $R$.

On trouve alors une anti-chaîne $(\mathbf{c}_i^k)_{k\leqslant n,\, i\in\mathbb{N}}$ telle que $B = (\mathbf{c}_i^k \cup \mathbf{x}_i)_{k\leqslant n,\, i\in\mathbb{N}}$ forme un ensemble de degrés calculatoirement indépendants (on montre sans peine qu'une telle anti-chaîne existe, par extensions finies). Pour $k \leqslant n$ fixé, soit $C_k = (\mathbf{c}_i^k)_{i\in\mathbb{N}}$. On définit finalement

$$S = \{\mathbf{c}_{i_1}^1 \cup \mathbf{x}_{i_1} \cup \cdots \cup \mathbf{c}_{i_n}^n \cup \mathbf{x}_{i_n} \ : \ (\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_n}) \in R\}.$$

Comme $B$ est calculatoirement indépendant, il existe pour tout $\mathbf{a} \in S$ un $n$-uplet de degrés $\mathbf{b}_1, \ldots, \mathbf{b}_n \in B$, unique à l'ordre près, tel que

$$\mathbf{a} = \mathbf{b}_1 \cup \cdots \cup \mathbf{b}_n.$$

Par ailleurs, l'indépendance calculatoire de $B$ garantit également que pour le degré $\mathbf{b}_1$ il existe un unique $i \leqslant n$ tel que $\mathbf{b}_1 = \mathbf{c}_m^i \cup \mathbf{x}_m$ pour un certain $m$, et ce $m$ est lui aussi unique. Il en va de même pour $\mathbf{b}_2$, $\mathbf{b}_3$, …, ce qui permet de définir $R$ de la manière suivante : $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in R$ si

$$\mathbf{x}_1 \leqslant \mathbf{b} \wedge \cdots \wedge \mathbf{x}_n \leqslant \mathbf{b} \wedge \exists \mathbf{y}_1 \in C_1 \ \ldots \exists \mathbf{y}_n \in C_n \ (\mathbf{x}_1 \cup \mathbf{y}_1) \cup \cdots \cup (\mathbf{x}_n \cup \mathbf{y}_n) \in S.$$

Comme chaque $C_i$ et comme $S$ sont des anti-chaînes, elles sont définissables d'après le lemme 4.6. Une telle formule est donc définissable dans les degrés Turing. ∎

Slaman et Woodin ont utilisé leur technique de codage comme point de départ à une étude complexe de la *rigidité* des degrés Turing. Une structure est dite *rigide* si elle n'admet pas d'automorphisme autre que l'identité, c'est-à-dire dans le cas des degrés, de bijection on triviale $f : \boldsymbol{\mathcal{D}} \to \boldsymbol{\mathcal{D}}$ telle que $\mathbf{a} \leqslant \mathbf{b} \leftrightarrow f(\mathbf{a}) \leqslant f(\mathbf{b})$. Par exemple, la structure $(\mathbb{R}, +, \times, \leqslant)$ des réels est une structure rigide. Étant donné un automorphisme $f : \mathbb{R} \to \mathbb{R}$, on doit avoir $f(0) + f(1) = f(1)$ et donc $f(0) = 0$, puis $f(1) = f(1) \times f(1)$

et donc $f(1) = 1$. En utilisant $f(n+1) = f(n) + 1$, on montre que $f$ est nécessairement l'identité sur $\mathbb{N}$, et l'on montre la même chose sur $\mathbb{Q}$ en jouant de la multiplication. Comme $f$ conserve les carrés, $f$ est croissante. On utilise alors pour terminer le fait que tout nombre réel est limite d'une paire de suites adjacentes de nombres rationnels, si bien que $f$ est forcément l'identité sur $\mathbb{R}$.

Peut-on faire quelque chose de similaire dans les degrés Turing ou bien peut-on « échanger » deux degrés $\mathbf{a}$ et $\mathbf{b}$, et étendre cet échange de manière consistante en un automorphisme sur $\boldsymbol{\mathcal{D}}$ ? La question reste pour le moment ouverte, même si Slaman et Woodin en ont, comme on va le voir derechef, considérablement réduit les possibilités.

**Théorème 4.8 (Slaman et Woodin [207])**
*Tout automorphisme sur les degrés Turing est l'identité sur* $\boldsymbol{\mathcal{D}}(\geqslant \mathbf{0}'')$.

Slaman et Woodin utilisent ce résultat pour montrer dans le même article que la fonction de double saut est définissable dans les degrés Turing. Ce résultat sera étendu plus tard par Shore et Slaman.

**Théorème 4.9 (Shore et Slaman [200])**
*Le saut Turing est définissable dans les degrés Turing, sans paramètres.*

Ce résultat peut alors à son tour être utilisé pour montrer que tout automorphisme sur les degrés Turing est l'identité sur $\boldsymbol{\mathcal{D}}(\geqslant \mathbf{0}')$.

La question suivante reste toutefois ouverte.

**Question 4.10.** Existe-t-il un automorphisme autre que l'identité sur les degrés Turing ? ⋆

## 5. Structure des degrés c. e.

Un autre champ de recherche important dans les degrés Turing consiste à en restreindre l'étude à un sous-ensemble. Dans cet ordre d'idée, l'étude de la structure $(\mathcal{R}, \leqslant)$ des degrés c. e. est certainement la plus développée. Nous faisons ici un résumé rapide des principaux résultats les concernant.

Commençons tout de suite par l'impressionnant théorème de densité de Sacks, qui se montre au terme de ce qui est certainement l'une des méthodes de priorité à blessure infinie les plus complexes.

**Théorème 5.1 (Sacks (1964))**
*L'ordre $(\mathcal{R}, \leqslant)$ est dense : étant donné des ensembles c. e. $A <_T B$, il existe un ensemble c. e. $C$ tel que $A <_T C <_T B$.*

La structure des degrés c. e. apparaît donc bien différente de celle des degrés Turing. Il y a tout de même des similarités : on vérifie ainsi sans problème que toute paire d'éléments admet une borne supérieure, l'ensemble $A \oplus B$ étant c. e. si $A$ et $B$ sont tous les deux c. e. Tout comme dans les degrés Turing, on peut montrer —en élaborant sur la construction de deux degrés incomparables— qu'il existe un ensemble dénombrable de degrés c. e. calculatoirement indépendants [164]. Tout comme pour les degrés Turing, on peut en déduire que tout ordre partiel dénombrable se plonge dans les degrés c. e., ce qui montre comme l'a remarqué Sacks que la théorie $\Pi^0_1$ des degrés c. e. est décidable.

**Théorème 5.2 (Sacks [188])**
*La théorie $\Pi^0_1$ des degrés c. e. est décidable.*

Étant donné une formule existentielle sur les degrés c. e., il suffit de vérifier si elle est compatible avec les axiomes d'un ordre partiel. Si tel est le cas elle est vraie, sinon elle est fausse.

Tout comme pour les degrés Turing, la question du plongement n'est pas vraiment satisfaisante en soi. Une question plus intéressante est celle du plongement de treillis, tel que les bornes supérieures sont envoyées vers les bornes supérieures, et les bornes inférieures sur les bornes inférieures. Les plongements de treillis mentionnés dorénavant seront considérés comme vérifiant cette contrainte. L'existence d'une paire minimale de degrés c. e. montre par exemple que le treillis en diamant généré par $\mathbf{c}_0, \mathbf{c}_1$ incomparables (c'est-à-dire avec $\mathbf{c}_0 \cap \mathbf{c}_1 < \mathbf{c}_0,\ \mathbf{c}_1 < \mathbf{c}_0 \cup \mathbf{c}_1$) peut se plonger de cette manière dans les degrés Turing, la borne inférieure étant le degré 0.

La construction effective d'une paire minimale de degrés c. e. (voir le théorème 13-5.2) a été exploitée pour montrer des résultats bien plus forts. Un treillis est dit *distributif* si $a \cap (b \cup c) = (a \cup b) \cap (a \cup c)$.

**Théorème 5.3 (Thomason [223] Lachlan et Lerman)**
*Tout treillis distributif dénombrable peut se plonger dans les degrés c. e.*

En ce qui concerne les treillis non distributifs, certains peuvent se plonger dans les degrés c. e. [131] et d'autres pas [134]. Aucune caractérisation n'est connue à ce jour.

Nous avons vu que la théorie des degrés Turing est de complexité maximale, il en va de même pour celle des degrés c. e. qui a la même complexité que la théorie constituée des formules vraies de l'arithmétique du premier ordre. Le premier résultat dans cette direction a été obtenu par Harrington et Shelah [83], qui ont montré que la théorie du premier ordre des degrés c. e. était indécidable. La preuve fut ensuite simplifiée et améliorée par Harrington et Slaman, puis par Nies, Shore et Slaman [168], qui ont montré le théorème suivant.

**Théorème 5.4 (Nies, Shore et Slaman [168])**
*On peut transformer effectivement un énoncé $F$ de l'arithmétique du premier ordre en énoncé $F^*$ sur les degrés c. e., tel que :*
*$F$ est vrai dans $(\mathbb{N}, +, \times, 0, 1)$ si, et seulement si, $F^*$ est vrai dans $(\mathcal{R}, \leqslant)$.*

Pour finir, Lempp, Nies et Slaman [137] ont montré que la théorie $\Pi^0_3$ des degrés c. e. était déjà indécidable, ce qui laisse ouverte la question suivante.

**Question 5.5.** La théorie $\Pi^0_2$ des degrés c. e. est-elle décidable ? ⋆