Université de Montréal

La logique ordinale de Turing

par
Benoit Potvin

Département de philosophie
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès arts (M.A.)
en Philosophie

Août, 2012

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

La logique ordinale de Turing

présenté par:

Benoit Potvin

a été évalué par un jury composé des personnes suivantes:

François Lepage,	président-rapporteur
Yvon Gauthier,	directeur de recherche
Jean-Pierre Marquis,	membre du jury

Mémoire accepté le: . . . . . .

RÉSUMÉ

Le sujet visé par cette dissertation est la *logique ordinale de Turing*. Nous nous référons au texte original de Turing *Systems of logic based on ordinals* (Turing [1939]), la thèse que Turing rédigea à Princeton sous la direction du professeur Alonzo Church. Le principe d'une *logique ordinale* consiste à *surmonter* localement l'incomplétude gödelienne pour l'arithmétique par le biais de progressions d'axiomes récursivement consistantes. Étant donné son importance considérable pour la théorie de la calculabilité et les fondements des mathématiques, cette recherche méconnue de Turing mérite une attention particulière. Nous retraçons ici le projet d'une logique ordinale, de ses origines dans le théorème d'incomplétude de Gödel jusqu'à ses avancées dans les développements de la théorie de la calculabilité. Nous concluons par une discussion philosophique sur les fondements des mathématiques en fonction d'un point de vue finitiste.

Mots clés : Incomplétude, calculabilité, indécidabilité, diagonalisation.

ABSTRACT

The main subject of this dissertation is *Turing's ordinal logic*, i.e. Turing's attempt to locally overcome Gödel's incompleteness by means of transfinite recursive progressions. We shall refer to the original 1939 text *Systems of logic based on ordinals* which is, in fact, Turing's Ph.D thesis at Princeton University under the direction of Professor Alonzo Church. Considering its importance for the theory of computability and the foundations of mathematics, Turing's paper certainly didn't get enough attention in the literature. Therefore, we want to retrace Turing's project of an ordinal logic from its very foundation in Gödel's incompleteness theorem to its further development in calculability theory. A discussion on the foundations of mathematics from a computational point of view will conclude this memoir.

Keywords: Incompleteness, calculability, undecidability, diagonalization.

TABLE DES MATIÈRES

LISTE DES SIGLES

Fbf(s)	Formule(s) bien formée(s)
$N.T.$	Number theoretic
TOT	Ensemble des fonctions totales
Conv	Est convertible en

NOTATION

$\mathbb{N}$	Ensemble des nombres entiers
$\mathbb{R}$	Ensemble des nombres réels
ω	Premier ordinal infini
$\forall$	Quantificateur universel
$\exists$	Quantificateur existentiel
$\in$	Élément de
$\mid$	Tel que
Σ	Alphabet d'entrée
Σ_i^0	Proposition du premier ordre en forme prénexe débutant par $\exists$
Π_i^0	Proposition du premier ordre en forme prénexe débutant par $\forall$
$\aleph_0$	Premier cardinal infini et cardinal de l'ensemble $\mathbb{N}$
$2^{\aleph_0}$	L'ensemble des parties de $\mathbb{N}$ et cardinal de l'ensemble $\mathbb{R}$

REMERCIEMENTS

Je tiens à remercier, de manière sincère et inéluctable, ma famille pour l'amour accordé, M. Gérard Desautels pour son aide *ex gratia* quant à ce mémoire mais aussi dans la vie – aussi petite soit-elle –, Gould pour son interprétation des *Variations Goldberg* de 1981 et le professeur Yvon Gauthier pour avoir gentiment accepté de diriger ce travail.

AVANT-PROPOS

La *philosophie des mathématiques* et l'*informatique* traitent de problèmes communs dans des langages différents. Pourtant, d'un côté, on ne s'intéressera qu'exceptionnellement aux résultats concrets et pratiques de la théorie de la calculabilité, tandis que de l'autre, on n'abordera que très rarement, sinon de manière superficielle, la démonstration du théorème d'incomplétude de Gödel et ses répercussions sur le programme de Hilbert. Le présent travail veut outrepasser ces différences théoriques en abordant d'une manière linéaire un texte séminal de la littérature mathématique, *Systems of logic based on ordinals* (Turing [1939]), qui se trouve être à la croisée des mathématiques, de l'informatique et de la philosophie. L'analyse de Turing est pour le moins complexe (*cum-plexus*) dans la mesure où il y a *entrelacement* des notions. Il ne s'agit certes que d'un mémoire et nous ne réinventons évidemment pas la roue. Nous n'apportons pas non plus de nouvelles connaissances. Nous ne faisons que des liens. Ceci dit, j'espère fournir au lecteur un exposé *constructif* sur le phénomène d'incomplétude de manière à justifier notre total désintérêt envers celui-ci. Nous ne devons plus s'ébahir devant un tel phénomène puisqu'il y a déjà plus de 70 ans que nous avons épuisé toute considération à son sujet. Les recherches contemporaines concernent plutôt la théorie de la complexité, où un travail considérable doit être accompli. À cet égard, une philosophie de l'informatique s'impose, mais il s'agit bien là, du travail d'une vie.

Muß es sein ? Es muß nicht sein !

CHAPITRE 1

INTRODUCTION

Il y a 100 ans naissait Alan Turing (23 juin 1912 – 7 juin 1954), un important mathématicien qui allait fonder le domaine de l'informatique et ainsi révolutionner le monde des sciences. Les mathématiques étant partie constituante de toute science – l'instrument qui sert d'intermédiaire entre la théorie et la pratique, entre la pensée et l'observation, est les mathématiques[1] – on oublie parfois que l'informatique en est en fait un sous-domaine. Ce qu'on appelle les mathématiques discrètes (ou mathématiques finies) sont au centre des théories de la calculabilité et de la complexité, les deux grands axes de l'informatique théorique. Il n'est donc pas surprenant que, d'une part, les recherches qui ont mené à la création de l'informatique concernent principalement les fondements des mathématiques et que, d'autre part, les limites actuelles de l'informatique peuvent et doivent influencer la pratique du mathématicien.

À regarder les différents ouvrages sur les fondements des mathématiques, on peut se questionner sur la *portée philosophique* de tels travaux. Or, s'il y a une tâche qui appartient bel et bien à la philosophie, c'est celle de l'étude des fondements de la connaissance et, conséquemment, de nos différentes sciences. Il est donc important de s'interroger sur la nature de nos outils et de nos méthodes, car ce n'est pas seulement par le fruit qu'on

[1] ≪Das Instrument, welches die Vermittlung bewirkt zwischen Theorie und Praxis, zwischen Denken und Beobachten, ist die Mathematik.≫ (Ma traduction. Hilbert [1930]).

connaît l'arbre[2] mais bien, aussi, par ses racines et sa structure. Un moment important de l'histoire des mathématiques est certes celui où Gödel établit ses résultats sur l'incomplétude d'une grande classe de systèmes formels. Le cerveau humain est fait tel que notre pensée est systématique et, par conséquent, nos théories le sont aussi. Le théorème d'incomplétude de Gödel ne concerne toutefois pas les théories physiques, biologiques, ou sociales puisque pour ce faire, il faudrait établir une isomorphie, c'est-à-dire une parfaite correspondance formelle, entre le système formel de l'arithmétique récursive et les différents modèles physiques. Or, tel n'est pas le cas. Les mathématiques fournissent des outils aux autres sciences et à défaut de croire que Dieu a créé le monde avec les nombres entiers – ou comme Scott Aaronson le prétend avec les nombres complexes (parce qu'ils sont algébriquement clos) – il serait farfelu d'étendre le théorème d'incomplétude à toute l'entreprise de la connaissance humaine. Une réflexion sur le théorème d'incomplétude et la logique ordinale de Turing nous permettra de circonscrire les différentes explications *mathématiques* qui justifient la présence de problèmes indécidables dans les théories actuelles. Le sujet visé de ce mémoire est la logique ordinale de Turing. Tous les sujets abordés sont donc introduits de manière à faciliter notre analyse du texte de Turing. Pour ce faire, nous procéderons de manière à rendre les faits tels qu'ils ont été historiquement, de l'incomplétude gödelienne (chapitre 2) au développement de la théorie de la calculabilité (chapitre 3), pour ensuite passer de la logique ordinale de Turing (chapitre 4) à une discussion critique sur les fondements

[2] ≪Ou rendez l'arbre bon et son fruit bon, ou rendez l'arbre mauvais et son fruit mauvais ; car c'est par le fruit qu'on connaît l'arbre.≫ Matthieu 12 :33.

des mathématiques (chapitre 5). Par ce mémoire, je souhaite réactualiser une recherche méconnue de Turing, celle sur les logiques ordinales, et souligner le caractère innovateur d'un grand homme qui ne fut peut-être pas de son époque.

> ≪C'est l'après-demain seulement qui m'appartient. Certains naissent post-humes.≫
>
> F. Nietzsche, Avant-propos, L'Antéchrist. 1895.

CHAPITRE 2

LE THÉORÈME D'INCOMPLÉTUDE

Le théorème d'incomplétude de Gödel (Gödel [1931]) est, dans l'histoire de la logique et des mathématiques, un des résultats les plus importants qui soient, sinon le principal résultat métathéorique négatif, cela ne fait guère de doute. Mais s'il fut utilisé à toutes les sauces, de la physique aux sciences sociales, et même jusqu'à la théologie, d'autres ont préféré limiter sa portée aux seuls fondements des mathématiques. Il existe de nombreux ouvrages généraux et autant accessibles qui concernent le théorème de Gödel (Ladrière [1957], Franzén [2005], Feferman [2006], Rosser [1939], Smullyan [2001]) et, par conséquent, nous ne nous occuperons pas ici de définir la portée *générale* du théorème. Nous tâcherons plutôt de circonscrire les raisons *mathématiques* qui établissent le phénomène d'incomplétude puisqu'il s'agit en fait de préparer le lecteur au sujet visé de ce mémoire : la logique ordinale de Turing. Ceci dit, nous affirmons toutefois qu'il n'affecte généralement pas le travail des mathématiciens, à savoir qu'il a très peu de conséquences sur les mathématiques en dehors du domaine de la logique. Dès lors, si, pour certains, le théorème d'incomplétude représente des bornes théoriques, ou garde-fous, utiles pour les recherches du mathématicien, comme le pense René Thom[1] (Thom [1983]), ou pour d'autres, qu'il affecte les théories jusqu'à leurs interprétations phy-

[1] Thom écrit à la page 25 : « Au-delà d'un certain intérêt philosophique – indéniable –, ces résultats ne font que démontrer qu'il est inutile de travailler dans certaines directions. C'est là – veuillez excuser le jeu de mots – leur utilité. Ce sont des sortes de garde-fous qui nous indiquent qu'il ne faut pas sortir de la route. ».

siques, comme le soutient Roger Penrose[2] (Penrose [1989], [1994]), il règne constamment dans le domaine des mathématiques un spectre sur les limitation internes des formalismes, de l'arithmétique de Peano à la théorie des ensembles de Zermelo-Fraenkel avec l'axiome du choix (ZFC). À défaut de croire aux fantômes, nous voulons porter une analyse assidue et critique sur quelques travaux importants relatifs aux fondements des mathématiques et, à cet effet, le théorème de Gödel représente un moyen fort adéquat d'introduire plusieurs notions élémentaires.

2.1 Le programme de Hilbert

S'il appartient à la philosophie de ≪*former, d'inventer, de fabriquer des concepts*≫(Deleuze et Guattari [1991]), alors il convient aussi au philosophe d'harmoniser dans un effort d'unité, au moins selon l'exigence de non-contradiction (i.e. la consistance ou la cohérence), les idées de sa pensée dans un système. Et bien que le critère de consistance soit au centre même de la connaissance humaine – *ex contradictione sequitur quodlibet* – il s'agit bien là d'une condition purement syntaxique qui n'impose aucune valeur de vérité. Pourtant, ce n'est pas tout à fait ce que David Hilbert (1862-1943), fameux mathématicien du vingtième siècle, croyait au sujet des mathématiques. Pour Hilbert, les êtres mathématiques, objets ou éléments idéaux, avaient une existence formelle fondée sur la seule notion syntaxique de consistance (Hilbert [1922]). Cette conception s'opposait ainsi à celle de Brouwer (1881-1966) et des intuitionnistes qui réduisaient

[2]Penrose a drôlement utilisé le théorème d'incomplétude dans une tentative de minimiser le potentiel technique relatif à l'intelligence artificielle. Pour une critique, lire Feferman [2011].

plutôt l'existence mathématique à sa *constructibilité*. Par ailleurs, le programme de Hilbert consistait à formaliser les mathématiques de manière à les structurer en un corps de formules démontrables. À cela devait s'ajouter une métamathématique, ou théorie de la démonstration, utile à l'élaboration des démonstrations selon le critère de non-contradiction. Hilbert écrit :

> ≪Nous voyons se confirmer ce qu'Aristote avait sans doute déjà pressenti : notre intelligence ne recourt nullement à de mystérieux artifices, elle procède au contraire selon des règles parfaitement déterminées que l'on peut formuler explicitement et qui constituent la garantie de l'objectivité absolue de son jugement.≫ (Hilbert [1930])

Le système formel représentait, en conséquence, un instrument destiné à rendre possible ces démonstrations. Hilbert était ainsi attentif aux critiques constructivistes de son époque. Le projet hilbertien cherchait à concilier les mathématiques classiques, incluant la théorie des ensembles transfinis de Cantor, et les importants soucis et considérations constructivistes des intuitionnistes (entre autres).

> ≪Nous nous proposons, partout où il y a seulement le moindre espoir d'en tirer quelque chose, de rechercher soigneusement les méthodes fécondes de définition et de raisonnement, de les soigner, de les étayer, et de les rendre utilisables. Du paradis que Cantor a créé pour nous, personne ne doit pouvoir nous chasser.≫ (Hilbert [1926])

disait-il avec conviction. Les intuitionnistes, s'appuyant sur une logique plus *faible* (ou justement *limitée*), devait nécessairement rejeter une partie importante de l'analyse classique. Hilbert, quant à lui, croyait certes possible de fonder constructivement l'analyse mais, aussi, de monter au paradis cantorien des ensembles transfinis et d'y redescendre, d'une manière finitiste, avec plus d'outils (i.e. plus d'*êtres* mathématiques) en sa possession. Mais à quel prix ? Il fallait dans un premier temps construire un système formel pour l'arithmétique et ensuite, de manière graduelle, l'étendre à toute l'analyse. L'arithmétique est donc la pierre angulaire du programme de Hilbert puisqu'au fondement de la consistance relative des systèmes formels repose la consistance de l'arithmétique. Ainsi, d'une part, il était nécessaire d'axiomatiser l'arithmétique et, d'autre part, de fournir une preuve de sa consistance.

2.2 L'arithmétique de Peano

On doit à Giuseppe Peano (1858-1932), un exposé axiomatique de la théorie des nombres entiers. Le système de Peano est composé des éléments suivants :

1. Un nombre illimité de variables individuelles ($X_0,\ X_1,\ \ldots$).

2. Une constante individuelle (0).

3. Une variable prédicative ϕ (qui représente une propriété quelconque d'entiers).

4. Un prédicat à un argument définissant la propriété *être un nombre entier*.

5. Deux prédicats à deux entiers définissant les propriétés de *successeur* et *d'égalité* (=).

Dans un ouvrage intitulé *Arithmetices principia, nova methodo exposita* (Peano, [1889]), Peano proposa un ensemble d'axiomes pour définir l'arithmétique. Il s'inspira d'un ouvrage de Dedekind (Dedekind [1888]) et on peut donc parler du système *Dedekind-Peano*. Les axiomes du système sont les suivants :

1. 0 à la propriété d'*être un nombre entier*.

2. Il n'y a aucun X_0 tel que X_0 ait la propriété d'*être un nombre entier* et que 0 soit le *successeur de* X_0. Autrement dit, 0 n'est le *successeur d*'aucun entier.

3. Si X_0 a la propriété d'*être un nombre entier* et que X_1 est son successeur, alors X_1 a aussi la propriété d'*être un entier*.

4. Si $X_1 = X_2$ et si X_3 est le *successeur de* X_1 et X_4 le *successeur de* X_2, alors $X_3 = X_4$.

5. Si 0 à la propriété ϕ et si, quel que soit le nombre entier X_1, il suffit que X_1 ait la propriété ϕ pour que son *successeur* l'ait aussi, alors quel que soit le nombre entier X_1, il a la propriété ϕ. C'est le principe d'induction, au premier ordre, aussi appelé schéma d'induction. Remarquons qu'il est exprimé au second ordre dans le texte de Peano.

La géométrie euclidienne, axiomatisée par Hilbert (Hilbert [1899]), contient 20 axiomes. Le système de Peano, illustré ci-haut, ne contient pas seulement 5 axiomes mais

bien une infinité dénombrable d'axiomes. Ainsi, et comme c'est le cas de la théorie des ensemble de Zermelo-Fraenkel ou de l'arithmétique de Presburger, les axiomes de l'arithmétique de Peano ne sont pas en nombre fini. C'est que l'axiome au premier ordre – $\forall x_1, \ldots, \forall x_n[(A(0) \wedge \forall x(A(x) \supset A(Sx))) \supset \forall x\, A(x)]$ – représente les propriétés comme des attributs d'individus dans la sémantique ensembliste où les individus correspondent aux nombres naturels de l'univers ensembliste de cardinalité $\aleph_0$. Bref, à partir du schéma d'axiomes de Peano, il devient possible de définir toutes les opérations (relations) arithmétiques classiques. Ceci dit, une preuve de la consistance de l'arithmétique de Peano, avant 1931, aurait pu sembler être une avancée considérable pour le programme de Hilbert. À cet égard, Gentzen (1909-1945) proposa en 1936 une preuve de consistance pour l'arithmétique de Peano (Gentzen [1936]). Cette preuve de Gentzen, nonobstant son caractère non-constructif du fait qu'elle fait appel à des concepts imprédicatifs (Gentzen a recours à l'induction transfinie sur les ω jusqu'à ε_0) s'inscrivait bel et bien dans le programme de Hilbert. Il s'agissait ainsi, pour Hilbert, d'un pas de plus pour les fondements *consistants* des mathématiques. Mais, avant même cette preuve de consistance par Gentzen, Gödel (1906-1978) établit un théorème important sur les limitations internes des formalismes qui allait freiner considérablement le projet hilbertien. La portée de l'instrument privilégié de Hilbert pour fonder les mathématiques, le système formel, devenait ainsi grandement limitée et son projet alors meurtri.

2.3 Le premier théorème d'incomplétude

Von Neumann (1903-1957) considérait Gödel comme le ≪plus grand logicien depuis Aristote≫ (Gödel [2003]). D'une part, von Neumann avait travaillé à l'élaboration du programme de Hilbert et, d'autre part, il avait été le seul mathématicien, parmi les Hans Hahn (1879-1934) et Rudolf Carnap (1891-1970), à saisir l'importance des considérations que Gödel fit, le 7 septembre 1930, sur l'incomplétude lors d'un tour de table post-conférences[3]. Ironiquement, cette journée là, Gödel allait présenter son théorème de complétude, sujet de sa dissertation sous la direction de Hahn[4]. Le théorème de complétude (Gödel [1929]) démontre la complétude *sémantique* de la logique du premier ordre. La logique du premier ordre (ou logique des connecteurs du premier ordre) est dite *complète* en ce sens que les axiomes et règles d'inférence utilisés permettent, dans un langage de premier ordre, de dériver toutes les conséquences logiques possibles. Ce théorème allait dans le même sens que le projet d'une théorie de la démonstration (métamathématique) de Hilbert dans la mesure où il garantissait la validité de l'appareil logique. Il justifiait, par la même occasion, l'utilisation de la méthode *algorithmique*[5]. Par ailleurs, le théorème d'incomplétude concerne plutôt la complétude *syntaxique* ou *négacomplétude*. Dans son texte de 1931 intitulé *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I.*[6], Gödel démontre son premier

[3]Pour plus d'informations à ce sujet, voir (Dawson [1991]).

[4]Il est intéressant de remarquer que l'extrait de la conférence par Hilbert cité dans la préface fait référence au 8 septembre 1930, et ce dans la même ville, à Königsberg.

[5]Voir le chapitre 3 à ce sujet.

[6]Une deuxième partie était prévue afin de démontrer plus explicitement le *deuxième théorème d'incomplétude* sur la consistance des systèmes formels.

résultat, aussi appelé *premier théorème d'incomplétude*. Il écrit (*théorème 9*) :

> ≪Il existe, dans tous les systèmes formels mentionnés dans le théorème 7[7], des problèmes indécidables [...]≫ (Gödel [1931])

Ce théorème de Gödel en est un sur la limitation interne des formalismes et s'attaque, par conséquent, directement au projet hilbertien d'exprimer les mathématiques en un système formel complet, consistant et décidable. La complétude (ou l'incomplétude) n'avait jamais vraiment été remise en question par Hilbert. Il découlait de sa conception consistante des êtres mathématiques que les mathématiques devaient être complètes et ses problèmes décidables. Hilbert écrit :

> ≪Pour nous, il n'y a pas d'*ignorabimus* et, selon moi, pas plus dans toutes les sciences de la nature. En opposition au naïf *ignorabimus*, je propose notre slogan : Nous devons savoir, nous saurons. ≫. (Ma traduction. Hilbert [1930])

Un système formel est dit complet *syntaxiquement* si pour toutes les formules bien formées (*fbfs*) closes A d'un système formel quelconque, on peut démontrer A ou la négation de A. Une *fbf* close (ou fermée) est un énoncé où il n'y a pas d'occurrence libre de variables. Le théorème d'incomplétude de Gödel stipule qu'il existe, pour une certaine classe de formalismes, au moins une proposition indécidable (ou deux avec sa

[7] À savoir les systèmes formels ω-consistants auxquels est ajouté une classe d'axiomes récursivement définissable.

négation). Une proposition A est dite *indécidable* – au sens de Turing – dans un système formel S, s'il est impossible de démontrer A ou $\neg A$ dans S.

Le système formel utilisé par Gödel pour le théorème d'incomplétude est, à peu de choses près, celui des *Principia Mathematica* de Russell (1872-1970) et Whitehead (1861-1947) (Russell et Whitehead [1928]) auquel sont ajoutés les axiomes de Peano. Il est, par conséquent, possible de représenter dans AP toutes les relations de la théorie des nombres. AP est, par définition, consistant puisqu'un système formel inconsistant est nécessairement complet. La proposition construite par Gödel affirme d'elle-même son indécidabilité (i.e. sa nature *indémontrable*) dans AP. Dans la présentation de son théorème, première partie du texte, Gödel reconnaît une certaine analogie entre le processus de construction qu'il utilise et le *paradoxe du menteur*. Le paradoxe du menteur s'énonce environ comme suit :

> «Épiménide le Crétois dit que tous les Crétois sont des menteurs».

Cet énoncé conduit immanquablement à une contradiction puisque si Épiménide, en affirmant que tous les Crétois mentent, dit vrai, alors lui-même qui est Crétois doit nécessairement mentir, ce qui rend la proposition fausse. Et vice-versa, d'où le paradoxe. Pour remédier à cette contradiction, il est nécessaire de considérer que l'affirmation d'Épiménide est énoncée dans une *métalangue*. Il devient alors possible, pour Épiménide, de dire la vérité d'un énoncé de premier niveau à partir d'un énoncé de deuxième niveau. De la même manière, nous remarquons que, dans ce travail, la *table des matières* figure elle-même parmi la table des matières. L'utilisation d'un langage for-

mel permet d'introduire cette distinction entre langage et métalangage et c'est ainsi que Gödel construisit sa proposition indécidable. Mais l'indécidabilité ne repose pas sur le caractère *autoréférentiel* de la proposition mais bien sur une application de la méthode *diagonale* de Cantor (1845-1918). En effet, il n'y a qu'une infime partie des problèmes indécidables qui ont un caractère autoréférentiel. Il existe toutefois une relation importante entre le *théorème de la récursion de Kleene* (ou le théorème du point fixe), l'autoréférentialité, la méthode diagonale et le théorème d'incomplétude. Ces nuances seront abordées au chapitre 3. Ceci dit, un système formel se présente, d'une part, comme un instrument permettant d'élaborer des théorèmes et, d'autre part, comme un objet dont on peut étudier les propriétés. À cet égard, Gödel établit une correspondance entre la métathéorie du système *AP* et l'arithmétique récursive. Il est, par ailleurs, maintenant nécessaire d'introduire la théorie des fonctions récursives.

2.3.1 Les fonction récursives

Les *fonctions récursives* de Gödel (Gödel [1931]) sont en fait, depuis la distinction faite par Kleene (1909-1994) dans un article intitulé *General Recursive Functions of Natural Numbers* [1936], des *fonctions récursives primitives*. Les fonctions récursives primitives se définissent à partir de trois fonctions initiales (aussi dites *de base*), à savoir :

1. les fonctions constantes, dont la fonction zéro :

$$Z(x) = c \text{ pour tout } x \text{ et } c \text{ une constante,}$$

2. la fonction successeur :

$$S(x) = x + 1 \text{ pour tout } x,$$

3. et les fonctions de projection (ou d'identité) :

$$\forall \vec{x} I_i^n(\vec{x}) = x_i \text{ ayant } n \text{ variables}$$

(où $\vec{x}$ représente $(x_1, \ldots, x_n)$)

Ces fonctions se construisent ensuite par un nombre fini d'applications de règles[8]

4. la règle de substitution (ou composition) :

$$f(x,\ y)\ =\ g(x,\ h(y)) \text{ et } f(\vec{x}) = g(h_m(\vec{x}_n))$$

(où g a originellement y_m pour variables auxquelles on substitue $h_1(\vec{x})$)

5. la règle de récursion :

$$f(\vec{x},\ 0)\ =\ g(\vec{x})$$

$$f(\vec{x},\ y+1)\ =\ h(\vec{x},\ y,\ f(\vec{x},\ y))$$

(où g est une fonction à n variables ; la fonction f à $n+1$ variables définie de cette façon est unique).

[8] Pour plus de détails, voir Gauthier [2010].

Toutes les fonctions récursives primitives sont des fonctions totales. Une fonction f de $\mathbb{N}^n$ dans $\mathbb{N}$, pour un nombre n d'arguments donnés, est dite *totale* si et seulement si elle a pour domaine l'ensemble de tous les n-tuplets ordonnés de nombres naturels. Autrement dit, la fonction totale f est définie pour chacune de ses entrées (arguments). Dans la théorie algorithmique où une preuve est considérée comme un ensemble *fini* de symboles, une fonction totale correspond à un objet *fini* (comme toute fonction calculable). Or, la classe des fonctions récursives primitives ne contient pas toutes les fonctions calculables. Les fonctions d'Ackermann (Ackermann [1928]) et de Sudan (Sudan [1927]) en sont des exemples. Par ailleurs, il est nécessaire d'élagir la classe des fonctions récursives primitives en introduisant un autre opérateur :

6. L'opérateur μ :

$$f(\vec{x}) = \mu y(g\ (\vec{x},\ y) = 0)$$

où μy signifie *le plus petit nombre y tel que* [...]. Autrement dit, de façon plus générale, pour un prédicat P, $\mu y[P(x_1,\ldots,x_n,y)]$ correspond au plus petit nombre y tel que $P(x_1,\ldots,x_n,y)$ est vrai. Du fait qu'un tel y n'existe pas toujours, l'opérateur μ joue un rôle analogue à l'intuition humaine dans la mesure où il permet d'effectuer une recherche de valeurs *non bornée* et de *sauter* par-dessus les valeurs non définies. Par conséquent, l'opérateur μ nous fait sortir du domaine des fonctions totales. Ainsi, la plus petite classe de fonctions qui :

1. contient les fonctions initiales,

2. est close par composition et récursion primitive,

3. et est close par le schéma de μ-récursion

est celle des *fonctions partielles récursives*. Les fonctions dites *récursives générales* sont des fonctions partielles récursives totales.

L'ensemble dénombrable des fonctions récursives primitives procède de manière constructive, comme le montre l'application des deux opérateurs de construction. Ainsi, toute construction à partir des fonctions initiales en fonction d'autres fonctions récursives est aussi récursive et, ainsi, par induction, il est possible de définir les fonctions de somme, de produit, de différence, etc. L'introduction de l'opérateur μ atténue cependant le caractère *constructif* mais permet de procéder, malgré tout, de manière *récursive*, à savoir qu'on limite l'opérateur μ (ou les quantificateurs universels et existentiels). Par conséquent, l'ensemble des fonctions primitives récursives et des fonctions récursives générales peut s'exprimer dans AP[9]. À cet égard, Gödel définit 45 relations (propriétés métathéoriques) en fonction de leur représentabilité dans le formalisme des fonctions récursives. Pour en énoncer seulement quelques-unes :

1. être une constante,

2. être une proposition élémentaire,

3. être une proposition obtenue par substitution à partir du schéma d'axiomes,

[9]Pour plus de détails sur le théorème de représentabilité et la fonction β voir les ouvrages de Gauthier [2010] et Franzén [2004].

4. être un axiome, etc.

Une quarante-sixième relation soulève notre attention :

> ≪46. $Bew(x) \equiv (Ey)\ y\ B\ x$
>
> x est une *formule démontrable*. [$Bew(x)$ est le seul des concepts 1-46 qui n'est pas récursif.]≫ (Ma traduction. Gödel [1931])

Cette propriété d'*être une démonstration d'une propriété* (ou dérivation) sera abrégée ici par $Dem(x)$ au lieu de $Bew(x)$, ce qui est plus *significatif* pour nous. La preuve de Gödel consiste à établir un paradoxe en fonction de l'impossibilité à définir récursivement cette propriété métathéorique dans *AP*. Pour ce faire, il est nécessaire de procéder à l'*arithmétisation* de la syntaxe de *AP*

2.3.2 L'arithmétisation de la syntaxe

L'arithmétisation de la syntaxe consiste à attribuer un nombre naturel unique à chaque symbole et suites de symbole, de manière à permettre une correspondance entre les énoncés intrathéoriques et métathéoriques de *AP* et l'arithmétique récursive. Pour procéder à cette arithmétisation de la syntaxe de *AP*, Gödel utilise une méthode qui découle du théorème fondamental de l'arithmétique. Ce théorème affirme que chaque entier strictement positif peut être écrit comme un unique produit de nombres premiers. La *numérotation de Gödel* associe donc à chaque composante primitive du langage de *AP* un nombre premier. Par exemple, supposons que :

1	correspond à	(
3	à	)
5	à	$\vee$
7	à	p
11	à	q
13	$\rightarrow$	$[\ldots]$

Pour obtenir le nombre de Gödel unique d'une expression formée de composantes primitives, il suffit d'attribuer respectivement le nombre correspondant de chaque signe à un nombre premier, sous la forme d'un produit. Dès lors, on peut attribuer un nombre de Gödel G à la formule $(p \vee q)$:

$$G = 2^1 * 3^7 * 5^5 * 7^{11} * 11^3.$$

À chaque symbole et suite de symboles du système *AP* correspond un nombre unique. Conséquemment, un nombre de Gödel étant donné, on peut retrouver la suite de symboles associée. Il s'agit d'un procédé récursif qui, en fonction de la notion d'*algorithme* (ou de la relation *Dem*), est calculable. Ainsi, on peut attribuer un nombre de Gödel à la démonstration d'un théorème. Un théorème est une proposition qu'il est possible de dériver, en un nombre fini d'étapes, à partir des axiomes et en fonction des règles d'inférence d'un système formel. Autrement dit, à chaque théorème est relié un algorithme. Chaque dérivation des théorèmes de *AP* peut alors se voir attribuer un nombre

unique par le processus de numérotation de Gödel. L'énoncé métathéorique *A est démontrable dans AP* peut donc, de manière hypothétique mais raisonnable, être exprimé sous la forme d'un énoncé arithmétique.

2.3.3 Construction d'une proposition indécidable

Il est nécessaire, pour étayer la preuve de Gödel, d'introduire une autre propriété métathéorique, notée *subs*. À cette relation métathéorique correspond une fonction récursive qui remplace une variable du premier type d'une proposition par un nombre entier. La proposition $subs[y(n),\ x,\ n]$ correspond au nombre de Gödel de la proposition obtenue en remplaçant, dans la formule y dont le nombre de Gödel est n, la variable libre x par n. Par exemple, à la formule bien formée $(x \vee y)$ appartient un nombre de Gödel n qu'il est possible de substituer à la place de x pour obtenir un nouveau nombre de Gödel correspondant à $(n \vee y)$. Ce nombre est donné par la relation $subs[y(n),\ x,\ n]$. Il faut aussi introduire l'opérateur de négation, ici $\neg$.

Soit $\neg Dem(x,\ n)$ signifiant *il n'existe pas de démonstration de la proposition x ayant le nombre de Gödel n*. On considère la proposition suivante : $\forall x \neg Dem[x,\ subs[y(x),\ x,\ n]]$, c'est-à-dire *pour tous les x, il n'existe pas de démonstration pour la proposition x ayant le nombre de gödel* $subs[y(x),\ x,\ n]$. En appliquant la relation *subs* à cette dernière proposition, on obtient la proposition indécidable suivante : $\forall x \neg Dem[x,\ subs[y(z),\ x,\ z]]$. Dit autrement, z est le nombre de Gödel de la proposition $\forall x \neg Dem[x,\ subs[y(z),\ x,\ z]]$. En substituant la variable x par le nombre de Gödel z, on obtient une proposition au

nombre de Gödel $subs[y(z),\ x,\ z]$ qui dit que la proposition ayant le nombre de Gödel $subs[y(z),\ x,\ z]$, à savoir *cette* proposition, est indémontrable. Ainsi, la proposition construite par Gödel – soit P, cette proposition – affirme d'elle-même son indécidabilité, à savoir qu'il est dit que P et $\neg P$ ne sont pas démontrables dans AP.

Il est, par ailleurs, nécessaire de supposer l'ω-consistance du système formel AP pour s'assurer de l'indécidabilité de P. L'ω-consistance de AP induit que pour toutes les fbfs $A(x)$ et tous les nombres naturels n (jusqu'à ω)

$$\vdash_{AP} A(\vec{n}) \rightarrow \nvdash_{AP} \exists x \neg A(x).$$

Dit autrement, l'ω-consistance implique que toutes les propriétés $P_i(n)$ où n est un nombre naturel de AP sont consistantes jusqu'à ω. Ceci dit, l'ω-consistance est une propriété plus forte que la simple consistance dans la mesure où celle-ci implique la consistance mais pas l'inverse (i.e. que la consistance n'implique pas l'ω-consistance). À cet égard, Kleene fournit un exemple d'une théorie consistante qui n'est pas ω-consistante. Une autre propriété, la Σ_1-correction (traduit de Σ_1-soundness) stipule qu'une théorie T est Σ_1-correcte si toutes les propositions de type Σ_1 (la hiérarchie arithmétique sera explicitée au chapitre 3) démontrables de T sont vraies dans le modèle standard de l'arithmétique. Par conséquent, la propriété d'ω-consistance utilisée par Gödel correspond exactement à celle de Σ_1-correction (ou 1-consistance) dans AP.

Ceci dit, il résulte que :

1. Si AP est cohérent alors la proposition P n'est pas démontrable dans AP.

2. Si AP est ω-consistant alors $\neg P$ n'est pas démontrable dans AP (voir Gauthier [2010] pour une preuve détaillée).

Rosser montra que l'ω-consistance peut se réduire, dans la preuve du théorème d'incomplétude, à la seule consistance (Rosser [1936]). La simple exigence de non-contradiction est primordiale et fondamentale pour un système formel, ce qui ajoute à la portée du théorème d'incomplétude. Nous traiterons, au chapitre trois, du théorème du point fixe de Kleene qui apporte des explications supplémentaires sur la démonstration du premier théorème d'incomplétude. Le théorème du point fixe permet de montrer qu'il est impossible de traduire dans le langage de l'arithmétique certaines propriétés métathéoriques. À cet égard, la propriété *AP est consistant* en est un autre exemple. Le deuxième théorème de Gödel énonce, en effet, que si AP est consistant, la *fbf* qui énonce cette propriété, notée Con_{AP}, est indémontrable dans AP. Une deuxième partie dédiée à cette preuve était initialement prévue par Gödel. Un mois environ après l'exposé de Gödel de 1930 à Königsberg, von Neumann lui écrivit une lettre concernant l'impossibilité de démontrer la consistance de AP dans AP, résultat que Gödel connaissait cependant déjà. Bref, Gödel n'eut pas besoin de publier sa preuve, le résultat étant généralement accepté. Dans les années qui suivirent, d'importants résultats permirent d'étudier la notion de *calculabilité relative* induite partiellement par le théorème d'incomplétude. Le théorème de Rice, par exemple, stipule que toute propriété *non triviale* de fonctions partielles est indécidable. Le chapitre trois se consacre à l'approfondissement du phénomène d'incomplétude.

CHAPITRE 3

LA THÉORIE DE LA CALCULABILITÉ

L'*Entscheidungsproblem*, ou problème de la décision, fut posé par Hilbert[1] et consiste à déterminer selon une méthode générale, c'est-à-dire à l'aide d'un *algorithme*, si une formule de la logique du premier ordre est ou n'est pas une formule valide. Le théorème d'incomplétude répond négativement à cette question mais d'une manière générale et non exemplifiée. Il revient plutôt à Church (Church [1936]) et Turing (Turing [1937]) d'avoir donné une réponse claire et négative au problème de la décision, le premier par ce qui est appelé le *théorème de Church* – et qui se rattache au problème des invariants de conversion dans le λ-calcul[2] – et le deuxième, par le problème de l'arrêt, en utilisant la notion de machine de Turing. Les machines de Turing, le λ-calcul et les fonctions récursives sont des formalismes équivalents quant à leur portée mais qui présentent des caractéristiques et atouts différents. Ainsi, par exemple, le formalisme des fonctions récursives traite de la théorie de la calculabilité en fonction de la composition de fonctions calculables à partir d'autres fonctions calculables (par induction), tandis que la notion de machine de Turing s'intéresse plutôt à la manière de *calculer* une fonction. Nous entendons par la théorie de la calculabilité (ou théorie de la récursivité) l'étude large des fonctions d'entiers. D'autres, comme Slaman (Slaman [2000]), préféreront mettre

[1] Pour une définition précise de l'*Entscheidungsproblem*, voir (Hilbert et Ackermann [1931]).

[2] Voir le chapitre quatre à ce sujet.

l'accent sur les liens qu'a la théorie de la calculabilité avec la théorie des ensembles pour parler d'une théorie de la *définissabilité*. Nul doute, Gödel initia une importante recherche sur la notion de *calculabilité effective* et nous rendons ici compte de quelques résultats importants[3].

3.1 Les machines de Turing.

Le formalisme des machines de Turing, introduit par Turing en 1936 (mais publié en 1937) définit la notion de calculabilité en fonction des capacités d'un dispositif – à savoir une *machine* – à *calculer*. À cet égard, les machines de Turing doivent au moins être capables de calculer l'ensemble des fonctions récursives. Pour ce faire, Turing imagine une machine composée des éléments suivants :

1. Un ruban *non-borné* vers la droite.

2. Un contrôleur.

3. Une tête de lecture et d'écriture.

Le ruban de la machine peut contenir un nombre dénombrable de cases contenant les symboles '0'ou '1'. La tête de lecture peut lire et écrire ces symboles dans n'importe quelle case du ruban. En fonction de ce qu'elle lit, la machine peut se diriger à gauche ou à droite ou simplement s'arrêter. Cette manipulation dépend du contenu de la case et de

[3]Pour un exposé des preuves détaillés de plusieurs théorèmes de la théorie de la calculabilité, voir (Sipser [2006]).

l'état courant de la machine. Une machine de Turing *M calcule* une fonction si pour une entrée donnée quelconque, M s'active (sur l'entrée) et finit par arrêter sa démarche. La méthode de calcul correspond à la séquence de symboles qui se trouve sur le ruban. Formellement, une machine de Turing (M) est un 7-uplets (Q, Σ, Γ, δ, q_0, $q_{acceptant}$, $q_{rejetant}$) définit de la manière suivante :

1. Q est un ensemble d'états,

2. Σ est l'alphabet d'entrée, c'est-à-dire un ensemble de symboles, ne contenant pas le symbole vide,

3. Γ est l'alphabet de sortie (sur le ruban) contenant le symbole vide et l'alphabet d'entrée.

4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ est la fonction de transition,

5. $q_0 \in Q$ est l'état initial (de départ),

6. $q_{acceptant} \in Q$, un état acceptant,

7. $q_{rejetant} \in Q$ est un état refusant, où $q_{refusant} \neq q_{acceptant}$.

Une machine de Turing est un ensemble fini d'instructions. Par ailleurs, l'ensemble des symboles de l'alphabet d'entrée et l'ensemble des états Q sont aussi finis. En fait, tout élément qui compose une machine de Turing standard est nécessairement *fini*, excepté le ruban qui est non borné dans le cas éventuel où le procesus de calcul ne se terminerait

pas. On peut, par ailleurs, considérer une machine de Turing comme un ordinateur ayant des ressources de temps et d'espace illimitées. On utilise communément le formalisme des machines de Turing en théorie de la calculabilité et l'ajout de contraintes de temps et d'espace (i.e. ce qui serait relatif à la *mémoire* de l'ordinateur) permet d'évaluer la *complexité* des calculs. Dès lors, certains problèmes dits *calculables* peuvent demander trop de ressources en temps ou espace comme c'est le cas, par exemple, du problème du *voyageur de commerce*.

Cela dit, une machine de Turing est un ensemble d'instructions qui permet de transiter d'une case à l'autre en fonction des états. C'est ce que la *fonction de transition* traduit par $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{G,D\}$, c'est-à-dire qu'à partir d'un état Q et d'un symbole du ruban (symbole $\in \Gamma$), la tête de lecture se déplace à gauche (G) ou à droite (D). Par définition, la tête de lecture se positionne à l'extrême gauche du ruban, à l'état q_0, et termine sur un état acceptant. Le ruban est initialement vierge. Le plus souvent, on utilise l'aphabet $\Sigma = \{0,\ 1\}$ puisque tout alphabet fini peut être codé à l'aide de seulement deux symboles. L'ensemble infini dénombrable Σ^* est celui de tous les mots de longeur arbitraire formés à partir des symboles de Σ. Un mot w dans Σ est une suite de symboles tel que $w = \{0,\ 1\}^*$[4].

Nous avons défini, au chapitre deux, les fonctions initiales du formalisme des fonctions récursives. Une machine de Turing peut facilement exprimer la fonction *zéro* dans la mesure où pour tout mot $w \in \Sigma^*$ donné en entrée, la tête commence la lecture à

[4]Autrement dit, un mot w est une chaîne de longueur arbitraire composée des seuls caractères de l'alphabet d'entrée.

l'extrême gauche du ruban, conserve ou imprime '1' sur la première case en fonction du premier symbole de w et imprime ensuite une série de '0' jusqu'au dernier symbole de w.

Si A est l'ensemble de toutes les chaînes de symboles qu'une machine de Turing M accepte, on dit que A est le langage de la machine M et nous écrirons $L(M) = A$. Un langage est un ensemble de chaînes de symboles. Une chaîne de symboles est une liste finie de symboles et un symbole est un membre d'un alphabet. Une machine M qui n'accepte aucune liste de symboles reconnait le langage vide $\varnothing$. Ceci dit, un langage qui est reconnu par une machine de Turing est dit Turing-reconnaissable. Ces langages sont tous *récursivement énumérables*. Ce concept peut se définir à partir de ce qu'on appelle un *énumérateur*. Un énumérateur est une machine de Turing configurée pour *énumérer* (ou générer) des chaînes de symboles :

1. L'entrée initiale est vide et seules les sorties sont prises en compte.

2. La tête de lecture ne revient jamais en arrière sur la bande de sortie ce qui fait que rien n'est effacé.

3. Tous les symboles utilisés, sauf le symbole *blanc* '♯' qui est utilisé comme séparateur des mots, appartiennent à l'alphabet Σ du langage L.

Si l'énumérateur ne s'arrête pas, il continuera indéfiniment d'imprimer sur le ruban. Un langage est dit récursivement énumérable (*r.e.*) si et seulement un énumérateur énumère tous les mots de l'alphabet. Dit autrement, un langage L sera *r.e.* si tous ses

mots sont énumérés au moins une fois sur la bande de sortie d'un énumérateur. L'ensemble vide est *r.e.* par définition. Dans le formalisme des fonctions récursives, nous dirons qu'un ensemble A est *r.e.* s'il existe une fonction f telle que : $A = Image(f) = \{y : \exists x[f(x) = y]\}$. Autrement dit, la notion de langage *r.e.* est analogue à celle de *fonction partielle récursive*, de la même manière que celle d'*ensemble récursif* peut être associée à la notion de *fonctions récursives*. On peut définir les ensembles récursivement énumérables comme les domaines des fonctions partielles.

Par ailleurs, il est important de définir le rapport entre *langage* et *ensemble*. Un langage est un ensemble où les mots du langage correspondent aux éléments de l'ensemble. Dès lors, tous les ensembles récursivement énumérables sont Turing-reconnaissables, c'est-à-dire qu'ils sont acceptés par une machine de Turing, Un langage *décidable* correspond à un ensemble *récursif*. Pour qu'un langage A soit dit décidable, il est nécessaire que A et son complément soient récursivement énumérable. Autrement dit, si un ensemble et son complément sont récursivement énumérables alors ils sont acceptés en entrée par une machine de Turing. Il sera alors possible, pour une entrée donnée, de simuler les deux langages en parallèle jusqu'à ce qu'une des deux machines s'arrête. Par conséquent, tout ensemble *r.e.* dont le complément est aussi *r.e.* est décidable. Un langage dont le complément n'est pas *r.e.* sera dit *semi-décidable*.

3.2 Le problème de l'arrêt de Turing

Bien que Gödel ait établi l'existence d'une proposition indécidable pour un grand ensemble de systèmes formels, il appartient à Turing d'avoir donné un exemple concret et algorithmique de l'insolubilité du problème de la décision dans son texte *On computable numbers with an application to the Entscheidungsproblem* (Turing [1937). Le problème de l'arrêt soulève l'impossibilité algorithmique de déterminer si une machine de Turing terminera sur une entrée donnée. Bien qu'on fasse habituellement référence au problème de l'arrêt, il s'agissait plutôt pour Turing du problème du non-arrêt, c'est-à-dire qu'il était impossible de déterminer si une machine allait procéder sans jamais s'arrêter[5].

Soit une machine de Turing $M = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_{acceptant},\ q_{rejetant})$ qui reconnaît le langage $L_A = \{\langle M,\ w\rangle \mid Macceptew\}$. Supposons qu'une machine de Turing standard décide L_A en fonction d'un algorithme hypothétique nommé *Accepte*. Dit autrement, nous supposons qu'il existe un algorithme permettant de dire si une Machine de Turing accepte un mot w. Pour continuer le parallèle entre les machines de Turing et les ordinateurs, on peut concevoir une *machine* comme un programme. Un programme est un algorithme. Dès lors, on peut dire que l'algorithme *Accepte* est une machine de Turing telle que :

[5]Contrairement aux formulations plus *contemporaines* du problème de l'arrêt, il n'y avait pas de paire dans la formulation originale de Turing.

$$Accepte(\langle M,\ w\rangle) = \begin{cases} accepte \text{ si } M \text{ accepte } w \\ refuse \text{ si } M \text{ refuse } w. \end{cases}$$

Précisons maintenant qu'une machine de Turing peut avoir comme entrée sa propre description. Cette instruction correspond à exécuter un programme sur lui-même. Il n'y a aucune ambiguïté à effectuer une telle action. Pour ce faire, on peut seulement concevoir une machine de Turing identique à une autre et simuler cette dernière sur la première. Pour les plus sceptiques, ajoutons qu'il est possible d'utiliser le théorème de la récursion de Kleene (ou théorème du point fixe) pour montrer qu'une machine de Turing peut imprimer sa propre description pour ensuite l'utiliser à des fins de calculs. C'est que le théorème du point fixe a, comme nous l'avons mentionné dans le chapitre deux, un caractère autoréférentiel, mais nous reviendrons sur cette idée[6].

On construit ensuite une nouvelle machine de Turing que nous appellerons volontairement *Diagonale*. Cette machine appelle *Accepte* et lui demande ce que M fait sur l'entrée M. *Diagonale* procède ensuite de la manière suivante :

$$Diagonale(\langle M\rangle) = \begin{cases} accepte \text{ si } M \text{ refuse } M \\ refuse \text{ si } M \text{ accepte } M. \end{cases}$$

La contradiction apparaît lorsqu'on donne à *Diagonale* l'entrée *Diagonale* :

$$Diagonale(\langle Diagonale\rangle) = \begin{cases} accepte \text{ si } Diagonale \text{ refuse } w \\ refuse \text{ si } Diagonale \text{ accepte } w. \end{cases}$$

[6]Von Neumann a d'ailleurs utilisé ce théorème dans la cadre des automates cellulaires autoreproducteurs (von Neumann [1963]).

Par conséquent, les deux algorithmes *Accepte* et *Diagonale* ne peuvent pas exister réellement du fait qu'ils utilisent le principes d'effectuer une manipulation en fonction de l'acceptabilité d'un mot par une machine de Turing. Ainsi, on établit l'indécidabilité du langage $L_A = \{\langle M,\ w\rangle \mid Maccepte\ w\}$. Pour poser précisément l'indécidabilité du problème de l'arrêt, c'est-à-dire l'indécidabilité du langage $Halte = \{\langle M,\ w\rangle \mid M$ arrête sur $w\}$, il suffit de la réduire à l'indécidabilité que nous venons de déterminer sur l'acceptation de w. Pour ce faire, on fait l'hypothèse qu'il existe une machine de Turing M_{Halte} qui décide le langage *Halte* et nous construisons à partir de M_{Halte} une autre machine de Turing qui décide L_A. Du fait que L_A est indécidable, on montre que *Halte* est, par conséquent, aussi indécidable.

3.3 La diagonalisation

La construction du problème de l'arrêt repose sur un processus de *diagonalisation*. C'est Georg Cantor (1845-1918) qui introduisit cette méthode par laquelle il créa notamment la *théorie des ensembles transfinis* (Cantor [1874]). La *cardinalité* d'un ensemble correspond à sa taille, c'est-à-dire aux nombres d'éléments de cet ensemble. Par conséquent, deux ensembles A et B ont la même cardinalité s'ils contiennent le même nombre d'éléments, à savoir que pour chaque élément appartenant à A on peut faire correspondre un élément unique appartenant à B. Cantor étendit cette méthode aux ensembles d'éléments infinis. À titre d'exemple, prenons l'ensemble des nombres naturels et l'ensemble des nombres premiers. Euclide nous a fourni une preuve constructive de

l'infinité des nombres premiers. Pourtant, pour chaque nombre premier, il est possible de faire correspondre un nombre naturel. Par exemple :

$$
\begin{aligned}
1 &\rightarrow 2 \\
2 &\rightarrow 3 \\
3 &\rightarrow 5 \\
4 &\rightarrow 7 \\
5 &\rightarrow 11. \\
6 &\rightarrow [\ldots]
\end{aligned}
$$

Ainsi, bien qu'il semble que la cardinalité de l'ensemble des nombres premiers soit plus petite que celle des nombres naturels, du fait que l'ensemble des nombres premiers est un sous-ensemble propre de l'ensemble des nombres naturels ($\mathbb{N}$), ils ont bel et bien la même cardinalité. Tout ensemble est dit *dénombrable* s'il est fini ou s'il a la même cardinalité que $\mathbb{N}$ puique chacun de ses éléments correspond à un nombre entier. De cette manière, on dénombre les éléments d'un ensemble infini. Le premier cardinal infini est noté $\aleph_0$ et c'est, par ailleurs, la cardinalité de $\mathbb{N}$.

Tous les ensembles infinis ne sont cependant pas dénombrables puisqu'il est impossible d'établir une bijection avec l'ensemble des nombres naturels. À cet égard, Cantor montre que l'ensemble des nombres réels $\mathbb{R}$ est non dénombrable. Un nombre réel est représenté par une partie entière et un nombre fini ou infini de décimales. Par conséquent,

il existe un nombre infini de nombres réels dans la mesure où il s'agit de toutes les combinaisons possibles à partir de l'ensemble infini des nombres naturels. Pour un ensemble E de cardinalité fini n, on appelle l'ensemble de toutes les combinaisons possibles à partir des éléments de E l'*ensemble des parties*, noté $P(E)$. L'ensemble des parties de E à la cardinalité 2^n. Mais qu'en est-il de la cardinalité de l'ensemble des parties d'un ensemble infini ? Pour répondre à cette question, Cantor utilise l'argument de la diagonale. Prenons une liste arbitraire des nombre réels de l'intervalle $[0,\ 1]$. Soit une relation f qui établit une correspondance entre un nombre naturel n et un nombre réel. Pour montrer que $\mathbb{R}$ ne peut pas être mis en bijection avec $\mathbb{N}$, il suffit de construire un nombre réel r tel que $\forall\ n,\ r\ \neq\ f(n)$. On construit un tel nombre r à partir de la n-ième décimale du nombre $f(n)$. Si la n-ième décimale de $f(n)$ est '0' alors la n-décimale de r sera '1' et si la n-ième décimale de $f(n)$ est différente de '0' alors la n-décimale de r sera '0'. Dans la mesure où le n-ième nombre peut ne pas avoir n décimales, nous écrirons '1' à la n-ième décimale de r. Le tableau suivant expose la méthode diagonale :

$$
\begin{aligned}
n = 1 &\ \rightarrow\ 0.\mathbf{1}2345\ldots \\
n = 2 &\ \rightarrow\ 0.0\mathbf{0}234\ldots \\
n = 3 &\ \rightarrow\ 0.40\mathbf{4}23\ldots \\
n = 4 &\ \rightarrow\ 0.230\mathbf{1}2\ldots \\
n = 5 &\ \rightarrow\ \quad [\ldots]
\end{aligned}
$$

Le nombre réel r débuterait alors ainsi : $r = 0.0100\ldots$. Conséquemment, le nombre

r ne peut jamais correspondre à un nombre de $f(n)$ puisque r est différent à au moins une décimale près de tout nombre $f(n)$. Par le processus de diagonalisation, nous obtenons un nombre $r \in [0, 1]$ qui ne correspond à aucun nombre $f(n)$ et qui démontre la non dénombrabilité de $\mathbb{R}$. Par ailleurs, la cardinalité de $\mathbb{R}$ est $2^{\aleph_0}$.

Le *théorème de Cantor* énonce que le cardinal d'un ensemble E est strictement inférieur à la cardinalité de l'ensemble de ses parties $P(E)$. On a vu précédemment que ce n'était pas le cas avec les sous-ensembles propres. La célèbre hypothèse du continu de Cantor, qui est le premier des 23 problèmes posés par Hilbert en 1900 (Hilbert [1902]), stipule que tout ensemble dont le cardinal serait strictement plus grand que celui de $\mathbb{N}$ est $\aleph_1$. Autrement dit, le deuxième cardinal infini serait $2^{\aleph_0}$, à savoir que $2^{\aleph_0} = \aleph_1$. L'hypothèse du continu est, d'après Gödel et Cohen (méthode du *forcing*), indécidable dans *ZFC*. Doit-on s'étonner ? Dans les prochains paragraphes, nous montrerons qu'au fondement des preuves d'indécidabilité repose toujours un argument de type diagonal.

3.4 Le processus diagonal du problème de l'arrêt

L'indécidabilité du problème de l'arrêt repose sur l'argument diagonal de Cantor. Turing connaissait le processus de diagonalisation et construisit l'indécidabilité du problème de l'arrêt à partir de celui-ci. Intitulée *Application of the diagonal process*, la huitième section du texte de Turing (Turing [1937]) montre bien cette démarche. Nous en résumons ici la technique.

Précisons d'abord qu'en fonction du théorème de Cantor, nous pouvons établir *a*

priori l'existence de langages indécidables, c'est-à-dire de langages qui ne sont pas récursivement énumérables. Le théorème de Cantor (sur la cardinalité de l'ensemble des parties d'un ensemble infini) montre qu'il existe un nombre indénombrable de langages. Autrement dit, l'ensemble de tous les langages est non dénombrable puisqu'il existe une infinité de langages infinis. Pareillement, il y a un nombre non dénombrable de fonctions non récursives. Tel est le cas de $\mathbb{R}$ puisqu'un langage ayant une infinité de mots infinis n'est pas dénombrable. Au contraire, l'ensemble de toutes les machines de Turing est dénombrable. De la même manière qu'il était possible, dans *AP*, d'attribuer un nombre de Gödel à un théorème, il est possible d'attribuer un nombre $n \in \mathbb{N}$ unique à chaque machine de Turing. C'est qu'une machine de Turing est un algorithme et donc il s'agit d'un objet fini. Par conséquent, l'ensemble de toutes les machines de Turing est un ensemble infini dénombrable de cardinalité $\aleph_0$. L'ensemble des fonctions récursives est aussi dénombrable puisqu'elles peuvent être codées avec un ensemble dénombrable de codes. Ajoutons qu'une machine de Turing ne peut reconnaître qu'un seul langage. Ceci dit, la cardinalité de l'ensemble des langages étant strictement plus grand que celle de l'ensemble des machines de Turing, il existe nécessairement des langages qui ne sont pas reconnus par des machines de Turing. Le parallèle ayant été fait entre les ensembles *r.e.* et les langages Turing-reconnaissables, en fonction de la notion d'*énumérateur*, tous les langages ne sont pas *r.e.* Dès lors, du fait qu'il existe des langages qui ne sont pas récursivement énumérables, on déduit qu'il existe des problèmes indécidables.

L'écrivain Alexandro Baricco exprime bien cette notion dans son roman *Novecento : un monologo*. Il écrit :

> ≪Imagine, maintenant : un piano. Les touches ont un début. Et les touches ont une fin. Toi, tu sais qu'il y en a quatre-vingt-huit, là-dessus personne ne peut te rouler. Elles sont pas infinies, elles. [*sic*] Et sur ces touches, la musique que tu peux jouer elle est infinie. […] Voilà ce qui me plaît. Ça c'est quelque chose qu'on peut vivre. Mais si je monte sur cette passerelle et que devant moi se déroule un clavier de millions de touches, des millions, des millions et des milliards de touches, qui ne finissent jamais, c'est la vérité vraie qu'elles ne finissent jamais, et ce clavier-là, il est infini. Et si ce clavier est infini, alors sur ce clavier-là, il n'y a aucune musique que tu puisses jouer. Tu n'es pas assis sur le bon tabouret : ce piano-là, c'est Dieu qui y joue. ≫. (Baricco [1994])

Si le piano infini de Novecento est celui de Dieu, la théorie des ensembles transfinis de Cantor lui appartient donc aussi, à Dieu. À cet égard, René Thom qualifie la construction cantorienne du transfinis de *cathédrale fantasmatique* (Thom [1983]).

Le même type d'argument peut être relié à la théorie des mondes possibles de Leibniz (Leibniz [1710]). Un monde est composé d'une infinité de monades. Si nous vivons dans le meilleur des mondes possibles, c'est que Dieu a su choisir, parmi l'infinité de mondes infinis, le meilleur d'entre eux. Par conséquent, seul Dieu peut jouer sur un piano infini

ou dénombrer un ensemble de cardinalité $2^{\aleph_0}$. Mais n'étant ni Dieu leibnizien, ni Dieu cantorien, revenons à notre fonction *monadique* diagonale :

$$Diagonale(\langle M \rangle) = \begin{cases} accepte \text{ si } M \text{ refuse } M \\ refuse \text{ si } M \text{ accepte } M. \end{cases}$$

ou dit autrement :

$$g(x) = \begin{cases} 1, \text{ si } f(x,x) = 0 \\ \text{non-défini, si } f(x,x) = 1. \end{cases}$$

Le raisonnement diagonal propre à cet algorithme devient apparent lorsqu'on analyse son comportement vis-à-vis la machine *Accepte* en fonction du processus de M sur M. Cette dernière est une machine de Turing quelconque parmi l'ensemble infini dénombrable de toutes les machines de Turing standards, que nous noterons M_i où $i \in \mathbb{N}$. Sur une entrée donnée, chaque machine M_i réagit évidemment d'une manière différente.

M_i / $\langle M_j \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	...
M_1	accepte	accepte	accepte	...
M_2		accepte	accepte	...
M_3		accepte		...
$\vdots$	accepte	accepte		$\ddots$

Ce tableau rend compte de l'acceptation de M_i sur la description $\langle M_i \rangle$. À chaque fois que la case est blanche, c'est que M_i rejette $\langle M_i \rangle$ ou n'arrête jamais sur celle-ci. Le prochain tableau rend compte du comportement de $Accepte(\langle M, M \rangle)$:

M_i / $\langle M_j \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	...
M_1	accepte	accepte	accepte	...
M_2	refuse	accepte	accepte	...
M_3	refuse	accepte	refuse	...
$\vdots$	accepte	accepte	refuse	$\ddots$

Le prochain tableau montre ce qui arrive quand *Diagonale* appelle *Accepte* :

M_i / $\langle M_j \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	...	$\langle Diagonale \rangle$	...
M_1	**accepte**	accepte	accepte	...	accepte	...
M_2	refuse	**accepte**	accepte	...	refuse	
M_3	refuse	accepte	**refuse**	...	accepte	...
$\vdots$				$\ddots$		
Diagonale	rejette	rejette	accepte		***Diagonale ?***	
$\vdots$			$\vdots$			$\ddots$

Dans la liste M_i apparaît évidemment *Diagonale* puisqu'il s'agit bien entendu d'une machine de Turing. Cette machine de Turing possède nécessairement une description, $\langle Diagonale \rangle$, qui elle aussi est dans M_j. La machine *Diagonale* ayant comme entrée *Accepte*$(\langle M,\ M \rangle)$, elle évalue donc tous les éléments où $i = j$, à savoir les éléments sur la diagonale de la matrice. *Diagonale* agit à l'inverse de *Accepte*. La contradiction survient lorsque *Diagonale* prend *Diagonale* comme entrée et tente de faire le contraire.

3.5 Diagonalisation et indécidabilité

Toutes les preuves d'indécidabilité utilisent l'argument diagonal d'une manière ou d'une autre. Si ce n'est pas de façon directe (comme, par exemple, le théorème de Church (Church [1936]) ou le problème de l'arrêt de Turing (Turing [1937]), c'est indirectement sous la forme du paradoxe du menteur d'Épiménide (comme le théorème d'incomplétude (Gödel [1931]) et le théorème de Post (Post [1944]). Le paradoxe du menteur implique certes la méthode diagonale mais tous les *objets diagonaux* ont, en fait, une saveur autoréférentielle. L'autoréférentialité inhérente au paradoxe du menteur est un cas particulier de l'utilisation de la diagonalisation à la Cantor. Soit une classe S d'objets récursivement énumérable (et donc dénombrable). Ces objets peuvent être des fonctions, des machines de Turing, des prédicats, etc. Soit P une propriété métathéorique qui fait correspondre à tout objet de S l'ensemble des entiers pour lesquelles la propriété P est vérifiée. P peut être la propriété *vraie*. Ceci dit, soit N une classe d'entiers tel qu'un entier n appartient à N si et seulement si cet entier ne fait pas partie du n-ième ensemble de la suite. L'ensemble E est, par ailleurs, un objet de la classe S. Soit r cet r-ième objet de S ($r \in \mathbb{N}$). Notons que r fait aussi partie de E. Dès lors, on obtient, en appliquant à r l'objet de la classe S associé à l'ensemble E, une proposition qui n'a pas la propriété P. Ce lien avec le paradoxe du menteur est dû au fait que r est, d'une part, utilisé dans un langage de premier ordre, à savoir r comme argument, et, d'autre part, utilisé dans un langage de second ordre, tel que E est le r-ième ensemble de S. Autrement dit, d'un côté le nombre r est l'argument d'une fonction, de l'autre il est le code

d'une fonction. La propriété P est construite en tant que point fixe dans S. De la même manière, la propriété *AP est consistant* joue aussi le rôle d'un point fixe. Le point fixe d'une fonction est une valeur qui ne change pas en fonction de l'application de la fonction. Le théorème du point fixe de Kleene (ou de récursion version *point fixe*) affirme que pour toute fonction récursive f, il existe un point fixe caractérisé par un nombre entier r (pour conserver la cohérence avec l'exemple donné ci-haut) tel que $\phi_{f(r)} = \phi_r$ (Kleene [1938]). C'est en fonction du caractère autoréférentiel de ce théorème qu'on a pu affirmer précédemment qu'une machine de Turing pouvait prendre en argument sa propre description. On peut ainsi établir l'existence de points fixes pour des fonctions qui sont des transformations calculables de machines de Turing. Autrement dit, on démontre qu'il existe des machines de Turing dont le comportement ne change pas en fonction de transformations. De cette manière, il est possible de simplifier maintes preuves en montrant la réduction d'un problème indécidable à la décidabilité du langage $L_A = \{\langle M,\ w\rangle \mid M\ accepte\ w\}$ utilisé dans la preuve de l'indécidabilité du problème de l'arrêt ci-haut. Ceci dit, l'indécidabilité se déploie par la diagonalisation autour de la construction de propriétés qui jouent le rôle de points fixes. À cet égard, le théorème de Rice (Rice [1953]) apporte d'importantes précisions :

> Soit A un ensemble d'indices tel que $A \neq \varnothing$ et $A \neq \mathbb{N}$ alors A est indécidable.

Autrement dit, le théorème de Rice établit que toute propriété non triviale est indécidable. Il est surprenant de voir que presque tous les ensembles d'indices sont indécidables. Une propriété P sur les langages est non triviale s'il existe au moins un langage *r.e.*

qui vérifie P et au moins un autre langage *r.e*. qui ne vérifie pas P. Ainsi, toute propriété qui n'est ni vraie pour tout langage ou ni fausse pour tout langage est indécidable. Par conséquent, on déduit, encore une fois, qu'il existe un nombre non dénombrable de langages indécidables tandis qu'il y a seulement un nombre dénombrable de langages décidables.

3.6 La hiérarchie arithmétique

La hiérarchie arithmétique de Kleene (Kleene [1943]) consiste à classifier les différents sous-ensembles de $\mathbb{N}$ en fonction des formules du calcul des prédicats qui leur sont associées. Pour ce faire, on doit mettre les propositions en forme prénexe – les formules du premier ordre sont toutes logiquement équivalentes à une formule de forme prénexe – c'est-à-dire que les quantificateurs se trouvent en leur forme réduite au début de la proposition (i.e. en fonction des blocs de quantificateurs consécutifs). Les classes Σ_n^m représentent les formules commençant par un quantificateur existentiel ($\exists$) et les classes Π_n^m représentent celles débutant par un quantificateur universel ($\forall$), pour $n,\ m\ \in\ \mathbb{N}$. Il existe une hiérarchie analytique pour les formules du second ordre et une autre, pour les formules du troisième ordre, etc. On distingue le *type* de la hiérarchie en fonction de l'exposant m. Pour un langage de premier ordre, $m = 0$. Puisque nous traiterons seulement de la hiérarchie arithmétique, nous omettrons souvent d'indiquer l'exposant. Les classes de la hiérarchie varient en fonction de l'alternance des quantificateurs comme le montre le tableau ci-dessous :

	n =1	n = 2	n = 3	...
Σ_n^0	$(\exists x_1)Q_1$	$(\exists x_1)(\forall x_2)Q_2$	$(\exists x_1)(\forall x_2)(\exists x_3)Q_3$	...
Π_n^0	$(\forall x_1)Q_1$	$(\forall x_1)(\exists x_2)Q_2$	$(\forall x_1)(\exists x_2)(\forall x_3)Q_3$	...

La classe Σ_0 (qui n'apparaît pas sur le tableau) représente les propositions à quantificateurs bornés de $\mathbb{N}$. À ce niveau, on ne différencie donc pas Σ_0 *de* Π_0. L'intersection des classes Σ_n et Π_n est notée Δ_n. La classe Σ_1 représente les ensembles récursivement énumérables et, de ce fait, la classe Δ_1 est précisément la classe de toutes les relations récursives. Toutes les classes de la hiérarchie ont les propriétés suivantes :

1. $\Delta_n \subset \Sigma_n$,

2. $\Delta_n \subset \Pi_n$,

3. $\Sigma_n \subset \Sigma_{n+1}$

4. $\Pi_n \subset \Pi_{n+1}$

5. $\Sigma_n \bigcup \Pi_n \subset \Delta_{n+1}, \forall n \geq 1$.

Nous aborderons au chapitre quatre le théorème de Post qui permet de relier la hiérarchie arithmétique aux degrés de Turing et qui permet de rendre ainsi compte de la décidabilité relative des propositions en fonction de leur degré de complexité (au niveau des quantificateurs). Nous avons maintenant une bonne partie des outils théoriques nécessaires pour l'étude du texte de Turing.

CHAPITRE 4

LA LOGIQUE ORDINALE DE TURING

Présenté comme étant *a profound and difficult paper* par Martin Davis (Davis [2004]), *Systems of logic based on ordinals* est la thèse que Turing rédigea à Princeton sous la direction d'Alonzo Church (Turing [1939]). Malgré ses apports considérables pour la théories de la calculabilité et les fondements des mathématiques, ce texte de Turing n'est que très rarement cité dans la littérature. Un ancien étudiant de Turing, Robin Gandy (1919-1995) écrivit dans une lettre à Max Newman (1897-1984)[1] en juin 1954 :

> ≪Alan considered that his paper on ordinal logics had never received the attention it deserved (he wouldn't admit that it was a stinker to read). ≫ (Copeland [2004])

Nous avons, jusqu'ici, introduit plusieurs notions théoriques empruntées à la théorie de la calculabilité qui faciliteront l'approche de ce texte. À cet égard, nous nous sommes déjà familiarisés, dans le chapitre deux, avec le formalisme des fonctions récursives et, au chapitre trois, avec celui des machines de Turing. Nous avons laissé savoir au lecteur qu'il s'agissait là de formalismes *équivalents*, en plus du λ-calcul de Church. Nous aborderons donc ce troisième formalisme puisqu'il est au centre même de la recherche menée par Turing sur les logiques ordinales.

[1] Newman était le professeur de Turing à Cambridge.

4.1 Le λ-calcul

Une des principales raisons qui explique l'impopularité du texte de Turing est le fait que celui-ci est entièrement couché dans le formalisme du lambda-calcul (λ-calcul). Ce formalisme fut développé par Church, peu après le résultat d'incomplétude de Gödel, dans l'espoir d'apporter de nouveaux fondements aux mathémathiques[2]. Church écrit :

> ≪[…] we present a set of postulates for the foundations of formal logic, in which we avoid use of the free, or real, variable, and in which we introduce a certain restriction on the law of excluded middle as a means of avoiding the paradoxes connected with the mathematics of the transfinite. ≫ (Church [1932])

Pour ce faire, Church développe une théorie *non typée* fondée sur le principe général qu'une fonction est une *règle*. Autrement dit, tout n'est que fonction et constructions de fonctions dans le λ-calcul. On appelle λ-*termes* les formules bien formées du formalisme. Il s'agit donc de *mots* construits à partir :

1. d'un ensemble de variables (x_0, x_1, …),
2. du symbole λ (qui représente l'*opérateur d'abstraction*),
3. de parenthèses.

[2] Voir (Barendregt [1981] pour un exposé détaillé du λ-calcul).

L'ensemble Λ des λ-termes se définit inductivement par :

1. $x \in \Lambda$ (i.e. une variable est un λ-terme),

2. Si $M \in \Lambda$ alors $(\lambda x M) \in \Lambda$ (i.e. les abstractions),

3. Si M et $N \in \Lambda$ alors (MN) $\in \Lambda$ (i.e. les applications),

où x est une variable arbitraire et M et N des λ-termes. Les formules du Λ-calcul n'ont donc de sens que dans leur construction et il nous appartient, ensuite, de leur donner celui (i.e. le sens) que nous voulons bien leur attribuer. Par exemple, nous pouvons définir la suite des nombres entiers de la manière suivante :

$$1 \rightarrow \lambda fx.f(x),$$

$$2 \rightarrow \lambda fx.f(f(x)),$$

$$3 \rightarrow \lambda fx.f(f(f(x))), \text{ etc.}$$

Les fonctions arithmétiques de bases peuvent, de même, être définies :

1. La fonction successeur : $\lambda\ n.\lambda\ f.\lambda\ x.f\ (n\ f\ x)$.

2. L'addition : $\lambda\ m.\lambda\ n.\lambda\ f.\lambda\ x.m\ f(n\ f\ x)$.

3. La multiplication : $\lambda\ m.\lambda\ n.\lambda\ f.m\ (n\ f)$.

On peut ensuite procéder à la définition de propriétés (i.e. de relations) en fonction des opérateurs de *conversions*. Un λ-terme M est dit *convertible* en un autre N (*M conv*

N) s'il existe une suite finie de *conversions* de *M* vers *N*[3]. À cet égard, Turing définit, par exemple, la *fbf Dt* telle que :

$$Dt(\mathbf{n},\mathbf{n})conv\ 3,$$

$$Dt(\mathbf{n}+\mathbf{m},\mathbf{n})conv\ 2,$$

$$Dt(\mathbf{n},\mathbf{n}+\mathbf{m})conv\ 1,$$

En 1935, Kleene et Rosser (Kleene et Rosser [1935]), deux étudiants de Church, montrèrent que le formalisme du λ-calcul était inconsistant en fonction du problème des invariants de conversions[4]. Autrement dit, il s'agissait de savoir s'il était possible de déterminer, par un algorithme, si, pour deux formules quelconques *M* et *N*, *M conv N*. De cela, il ressortit une situation paradoxale rappelant le paradoxe de Richard sur la définissabilité des nombres réels. Il devint alors possible de rendre compte de l'indécidabilité d'une manière différente de celle du théorème de Gödel et des fonctions récursives.

La thèse de Turing fait donc suite à l'échec du projet fondationnel de Church. En 1937, Turing quitta Newman et le King's College pour rejoindre Church à Princeton : ils avaient tous deux donné une réponse négative à l'*Entscheidungsproblem* mais dans des formalismes différents. La notion de machine développée par Turing avait cependant le mérite d'apporter une définition intuitive de la calculabilité qui n'avait pas toute la lourdeur du λ-calcul. En 1937, Turing montra dans un article intitulé *Computability*

[3] Voir le chapitre un de (Turing [1939]) pour une exposition des règles de conversions.

[4] Aczel montra que la théorie inconsistante de Frege contenait essentiellement le formalisme du λ-calcul (Aczel [1980]).

and Lambda-definability (Turing [1936]) l'équivalence des deux formalismes. Ceci dit, la thèse de Turing est entièrement couchée dans le formalisme du λ-calcul. Il ne faut toutefois pas croire qu'il s'agit là d'une exigence de Church. Il s'agit plutôt d'un choix stratégique, de la part de Turing, pour le développement d'une logique ordinale puisque le λ-calcul représente un moyen fort efficace pour passer de l'intuition à l'abstraction (et vice-versa). Nous reviendrons sur le rôle que peut jouer l'intuition pour les fondements des mathématiques puisqu'il est nécessaire, avant de poursuivre, de décrire le projet de Turing.

4.2 Le projet d'une logique ordinale

Dans sa thèse, Turing examine les possibilités de minimiser le phénomène d'incomplétude par le biais d'une logique ordinale. Pour Turing, une logique est un système formel. Il s'agit donc d'une tentative de réponse directe au théorème d'incomplétude de Gödel dans la mesure où celui-ci concerne précisément la notion de *système formel*. Turing débute sa thèse ainsi :

> ≪The well-known theorem of Gödel shows that every system of logic is in a certain sense incomplete, but at the same time it indicates means whereby from a system L of logic a more complete system L' may be obtained. By repeating the process we get a sequence $L, L_1 = L', L_2 = L'_1, [\ldots]$ each more complete than the preceding. A logic L_ω may then be constructed in which the provable theorems are the totality of theorems provable with the help of

> $L, L1, L2, [\ldots]$. We may then form L_{2_ω} related to L_ω in the same way that L_ω was related to L. Proceeding in this way we can associate a system of logic with any constructive ordinal. ≫. (Turing [1939])

Il s'agit d'obtenir une progression de systèmes formels, les uns plus complets que les précédents. Autrement dit, soit L_0 un système formel consistant et L_1 une extension de L_0, à savoir qu'ils partagent le même langage et règles d'inférence. L_1 est obtenu en ajoutant un axiome à L_0. Par exemple, on peut ajouter à L_0 l'axiome L_0 *est consistant* de manière à obtenir une extension L_1 plus complète sémantiquement que L_0. Par ce procédé, on peut *théoriquement* obtenir une progression de systèmes formels contenant toujours plus d'axiomes telle que $L_{n+1} = \cup L_n$ (où $n \in \mathbb{N}$).

Le projet de Turing succède ainsi au programme de Hilbert dans une tentative de reprendre et sauver l'idée du formalisme dans la quête fondationnelle des mathématiques. Cette recherche menée par Turing est pour le moins embryonnaire et s'intéresse avant tout, d'une manière générale, à la potentialité d'une telle méthode. Turing écrit :

> ≪The subject matter, roughly speaking, is constructive systems of logic, but since the purpose is directed towards choosing a particular constructive system of logic for practical use, an attempt at this stage to put our theorems into constructive form would be putting the cart before the horse.≫. (Turing [1939])

L'aspect *constructif* des logiques ordinales repose sur le principe de *progression récursivement consistante*. Pour ce faire, il est nécessaire d'utiliser une *notation* pour les ordinaux, à savoir un *système d'expressions* qui permet d'associer une expression à chacun des ordinaux inférieurs de manière constructive. Un tel système avait d'abord été élaboré par Church et Kleene dans le cadre du λ-calcul (Church et Kleene [1936]). Celui-ci, communément appelé *système O*, prend en compte tous les ordinaux de la première classe et une partie de ceux de la deuxième classe de Cantor. À cet égard, les ordinaux doivent seulement être constructifs, qu'ils soient des ordinaux du fini ou du transfini. Le plus petit ordinal non récursif est noté ω^{CK}[5]. Il s'agit de la limite théorique des logiques ordinales et nous comprenons, par ailleurs, qu'il n'y a de constructif que la méthode. Mais jusqu'où sera-t-il nécessaire de progresser pour obtenir un quelconque résultat de complétude ? S'il est nécessaire de poursuivre vers L_ω, $L_\omega + 1, L_\omega + 2, \ldots, L_\omega * 2, [\ldots]$, ou même jusqu'à ω^{CK}, on devra se questionner sur la portée d'un tel résultat de complétude. Turing écrit bien, ci-haut, qu'il veut construire un outil mathématique *praticable*. Il ajoute :

> ≪In consequence of the impossibility of finding a formal logic which wholly eliminates the necessity of using intuition, we naturally turn to non-constructive systems of logic with which not all steps in a proof are mechanical, some being intuitive. [...]. We want it to show quite clearly when a step makes use of intuition, and when it is purely formal. The strain put on the intuition

[5] *CK* pour l'ordinal *Church-Kleene*.

should be a minimum.≫. (Turing [1939])

Ainsi, le projet de Turing, étant fondé sur l'impossibilité même d'échapper de manière *finitiste* au théorème d'incomplétude de Gödel, repose, d'une part, sur l'hypothèse de faire appel à l'intuition mathématique et, d'autre part, sur la volonté de Turing à restreindre la complétude à une certaine classe de problèmes appelé *number-theoretic* (ou arithmétiques).

4.3 Les théorèmes arithmétiques ou théorèmes *Number-theoretic (N.T)*.

Dans son texte, Turing s'intéresse à une classe particulière de propositions arithmétiques qu'il appelle *number-theoretic (N.T.) theorems* ou théorèmes arithmétiques et qu'il définit de la manière suivante :

≪[...] for each natural number x there exists a natural number y such that $\phi(x,y)$ vanishes, where $\phi(x,y)$ is primitive recursive.≫ (Turing [1939])

Notons que Turing fait référence à un ensemble de fonctions bien précis qui ne correspond généralement pas dans la littérature à ce qu'on appelle les *number-theoretic functions* (ou fonctions arithmétiques), mais à un ensemble beaucoup plus restreint, c'est-à-dire les propositions de la forme $\forall x \, \exists \, y \; : \; [\phi(x, \, y) \; = \; 0]$, où $\phi(x, \, y)$ est une fonction récursive. Turing écrit :

≪I should assert that theorems of this kind have an importance which makes it worthwhile to give them special consideration.≫. (Turing [1939])

Si Turing accorde beaucoup d'importance à cette classe c'est parce qu'il présume que plusieurs problèmes de la théorie des nombres lui appartiennent, notamment l'hypothèse de Riemann et le dernier théorème de Fermat. La classe des *N.T.* théorèmes de Turing correspond aujourd'hui à un sous-ensemble de la classe Π_2 de la hiérarchie arithmétique de Kleene. Les problèmes Π_2 ont la forme générale irréductible $\forall x \; \exists \; y \; F(x, \; y)$, sans restriction précise. Or, l'hypothèse de Riemann et le dernier théorème de Fermat sont deux problèmes qui se rapportent plutôt à Π_1, c'est-à-dire aux propositions de la forme $\forall x \; F(x)$[6]. Cette tentative de réduire quelques problèmes à une seule et même classe représente, de la part de Turing, une idée ingénieuse et avant-gardiste. Heureusement, il n'y eu aucune répercussion quant à cette bévue sur le projet d'une logique ordinale. Les raisons sont simples : d'une part, le théorème d'incomplétude concerne justement les propositions de la classe Π_1, c'est-à-dire les propositions arithmétiques vraies mais indémontrables de la forme $\forall x \, F(x)$, et, d'autre part, Turing s'intéresse plus précisément aux théorèmes *N.T.* qui appartenaient effectivement à Π_2. Dès lors, si Turing obtenait la complétude pour Π_2, il l'obtenait aussi pour Π_1. Dans son article de 1937, Turing avait établi l'indécidabilité en fonction des machines dites *circle-free* (ou non circulaires) : c'est le problème de l'arrêt que nous avons abordé au chapitre précédent. Une machine est dite non circulaire si elle imprime sur son ruban de sortie une infinité de '0' ou de '1'. Turing écrit :

≪The behaviour of the machine may be described roughly as follows : the

[6]Il est certes difficile de réduire à sa plus simple expression un problème de manière à le classer dans la hiérarchie arithmétique puisqu'on peut difficilement être certain de l'irréductibilité de celui-ci.

> machine is one for the calculation of the primitive recursive function $\Theta(x)$ of the number-theoretic problem, except that the results of the calculation are first arranged in a form in which the figure 0 and 1 do not occur, and the machine is then modified so that, whenever it has found that the function vanishes for some value of the argument, then 0 is printed. The machine is circle free if and only if $\Theta(x)$ vanishes for infinitely many values of the argument.≫. (Turing [1939])

Il est intéressant de remarquer que le problème de déterminer si une machine M est non circulaire correspond, dans des termes actuels, au problème de savoir si une fonction f est totale (*TOT*). Une fonction est totale si elle est définie pour toutes les entrées, c'est-à-dire pour tout son domaine de définition. Autrement dit, une machine M appartient à *TOT* si et seulement si $(\forall y)$ [M_w s'arrête sur y] (c'est-à-dire que pour tout y il existe une machine qui s'arrête sur y quand on lui donne le mot w en entrée). On voit donc que le problème qui consiste à déterminer si une fonction est totale appartient à Π_2. Or, tous les problèmes *N.T.* de Turing se réduisent à un problème de *TOT*. Turing écrit :

> ≪We cannot say that the question about de truth of any number-theoretic theorem is reducible to a question about whether a corresponding computable function vanishes infinitely ; we should have rather to say that it is reducible to the problem of whether a certain machine is circle free and calculates an identically vanishing function.≫. (Turing [1939])

Par conséquent, si Turing obtenait la complétude pour les problèmes *N.T.*, il décidait aussi le problème de déterminer si une fonction est totale. Par le théorème de Rice, nous savons aujourd'hui que *TOT* est décidable. Par ailleurs, si *TOT* devenait calculable, l'argument diagonal ne serait alors plus valide et le problème de déterminer si une machine s'arrête ou non sur une entrée n'en serait plus un. Nous avons vu que toutes les preuves d'indécidabilité reposent sur un argument diagonal. On peut alors se demander, et surtout si nous étions à l'époque de Turing, si la décidabilité de *TOT* entraine l'éradication totale de l'indécidabilité. À cet égard, Turing fit un coup de maître.

4.4 L'oracle

C'est dans ses travaux sur les logiques ordinales que Turing pose les fondements de la calculabilité relative, c'est-à-dire la comparaison de la complexité des problèmes incalculables en introduisant le concept d'oracle. Dit autrement, certains problèmes sont plus indécidables que d'autres. La réduction ou réductibilité de Turing, outil nécessaire pour établir les degrés d'indécidabilité, sera ensuite définie par Post dans un important article de 1944, *Recursively Enumerable Sets of Positive Integers and their Decision Problems* (Post [1944]), à la suite des propos de Turing au paragraphe 4 de sa thèse *A type of problem which is not number-theoretic*. Après avoir établi la classe des problèmes *N.T.*, Turing voulut obtenir un problème qui n'appartenait pas à cette classe. De cette manière, Turing fixait la limite qu'il considérait atteignable par le biais d'une logique ordinale dans la mesure où l'oracle jouait le même rôle, ni moins fort ni plus fort, que l'intuition

mathématique nécessaire au projet d'une logique ordinale. Donc, au lieu d'utiliser une fois de plus l'argument diagonal, ce qui aurait été simple et efficace, Turing introduisit un concept original, celui d'oracle. Il écrit :

> ≪Let us suppose that we are supplied with some unspecified means of solving number-theoretic problems ; a kind of oracle as it were. We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine.≫. (Turing [1939])

Turing appellera *o-machine* une machine de Turing équipée d'un oracle. Intuitivement, disons que l'oracle peut répondre à toute question précise par *oui* (ou 1) et *non* (ou 0). En fait, la formalisation d'une telle machine se fait exactement de la même manière qu'une machine de Turing standard (c'est-à-dire qu'elle s'encode de manière standard), à l'exception du fait qu'elle est équipée d'une bande supplémentaire et de trois états spéciaux. Dès lors, l'oracle ne fait pas partie de la définition même de la *o*-machine. On peut concevoir la solution donné par un oracle comme un nombre réel écrit sur la bande. Toutefois, Turing montre qu'une *o*-machine ne peut pas décider le problème d'arrêt pour une autre o-machine tandis qu'elle peut évidemment résoudre n'importe quel problème *N.T.* particulier. On obtient donc un problème qui n'est pas *N.T.* Autrement dit, si *TOT* et le problème de l'arrêt étaient décidables, tous les autres problèmes ne deviennent pas nécessairement décidables. En fait, toute machine de Turing, de quelque type que ce soit, est incapable de décider son propre problème d'arrêt. Par conséquent, les *degrés de Turing* rendent compte de ces sauts en fonction de l'irréductibilité d'un problème

indécidable à un autre problème indécidable. Ainsi, bien que Turing ait cerné l'existence des différents degrés d'indécidabilité, il faudra attendre la hiérarchie arithmétique telle qu'énoncée par Kleene où, par exemple, on établit l'incomparabilité entre Σ_i et Π_i, et autres corollaires. À cet égard, le théorème de Post propose une façon de passer des degrés de Turing à la hiérarchie arithmétique. Pour ce faire, il faut utiliser la *réduction-Turing*, à savoir qu'un problème A est *Turing-réductible* à un problème B si, étant donné une o-machine solutionnant B, on peut aussi solutionner A. Dès lors, deux problèmes ont le même degré s'ils sont Turing-réductibles (ou inter-réductibles). Les programmes indécidables varient donc en complexité selon le nombre de sauts de Turing applicables. Ceci dit, cette anticipation de Turing est pour le moins remarquable et joua un rôle dominant dans sa recherche sur la logique ordinale.

4.5 Les formules logiques et ordinales

Une formule *logique* (ou simplement logique) a pour but de décrire un système formel particulier. Pour qu'une formule L soit une logique, L doit avoir la propriété que si $L(A)conv\ 2$ alors le λ-terme A est double. En se référant au sens *arithmétique* que nous avons attribué, plus tôt, aux λ-termes, on peut écrire $L(A)conv\lambda fx.f(f(x))$. *A est double* signifie que $A(x)$ est convertible à 2 pour tous les x. Dire que $A(x)$ est convertible à 2 signifie que pour chaque x, il existe un entier positif y tel qu'une propriété calculable $\theta(x,\ y)$ est vraie (ou pour chaque nombre naturel x il existe un nombre naturel y tel que $\theta(x,y)$ s'annule). Dès lors, tous les théorèmes *N.T.* équivalent, selon leur définition, à une

proposition de type $L(A)conv\ 2$ et, inversement, toutes ces propositions correspondent à un théorème *N.T.* Une formule logique donne alors un moyen de vérifier les théorèmes *N.T.* au moyen de leurs conditions de satisfaction. Ainsi, si L est une formule logique et $L(A)conv\ 2$, alors A est double et nous savons que la proposition *N.T.* correspondante est vraie. Si L est une logique, on ne peut toutefois pas avoir de conditions de satisfaction pour la vérité de toutes les propositions *N.T.* Par conséquent, on détermine l'extension de L comme formant l'ensemble des propositions A pour lesquelles $L(A)conv\ 2$. On couche alors notre progression récursivement consistante dans une notation pour les ordinaux.

Les λ-termes sont des formules ordinales dans la mesure où ils sont associés constructivement à des nombres ordinaux. Il n'existe pas de méthode canonique à cet effet et Turing utilise un système précis qui utilise six critères pour la détermination d'une formule ordinale, très proche du système O de Church et Kleene. Toute notation pour ordinaux doit cependant être partiellement ordonnée et, par ailleurs, définir un certain *zéro*, une fonction *successeur* et une *limite supérieure* (on procède par induction transfinie). Une logique ordinale est une *fbf* Λ telle que $\Lambda(\Sigma)$ est une formule logique à chaque fois que Σ est une formule ordinale. Autrement dit, on peut associer un nombre ordinal à toute dérivation d'une logique ordinale. Λ est une logique ordinale si toute dérivation de Λ (i.e. $\Lambda(\Sigma)$) est une formule logique. Une formule logique représente un système formel par une suite bien ordonnée de propositions. On associe à ces propositions un ordinal constructif. La suite de ces ordinaux forme alors un ensemble bien ordonné qu'on appelle une *logique*. Les travaux de Feferman résument bien cette idée dans un vocabulaire

plus actuel (Feferman [1988, 2006]) : $\Lambda = L_a$ tel que a est élément de O, c'est-à-dire que si A est une proposition Π_2 vraie, alors L_a démontre A pour un a élément de O. Or, le point est qu'il n'existe pas de procédure décisionnelle calculable (ou algorithmique), permettant de déterminer si un λ-terme est une formule ordinale (la situation serait évidemment la même avec l'utilisation des fonctions récursives) ou, dit autrement dans les termes de Kleene, O n'est simplement pas calculable. Turing connaissait, dès le départ, cette caractéristique particulière puisqu'il était aussi bien connu qu'aucune méthode générale permettait de déterminer si une formule du λ-calcul avait ou non une forme normale . Turing renvoie à Church ici :

> ≪There is (demonstrably) no process whereby it can be said of a formula whether it has a normal form.≫. (Turing [1939])

La détermination des formules ordinales est liée à celles de forme normale dans la mesure où si E représente une classe de formules ordinales et A n'importe quelle formule ordinale de cette classe, alors il n'existe aucune méthode par laquelle on peut déterminer si une *fbf* de forme normale appartient à E. Si Church met *demonstrably* entre parenthèses, c'est qu'il se rapporte à une autre procédure décisionnelle qui n'est toutefois pas mécanique, à savoir l'intuition. Church et Turing accordaient une grande place à l'intuition. Turing écrit :

> ≪Gödel's theorem shows that such system cannot be wholly mechanical ; but with a complete ordinal logic we should be able to confine the non-

> mechanical steps entirely to verification that particular formulae are ordinal formulae.» (Turing [1939])

Le caractère profondément abstrait du λ-calcul permet de circonscrire la place nécessaire à accorder à l'intuition afin de combler les *trous* creusés par l'incomplétude gödelienne. C'est dans cet esprit que le λ-calcul – l'opérateur λ est celui d'abstraction – est le formalisme le plus approprié aux fins d'une logique ordinale. Le théorème de Church établit l'inconsistance du λ-calcul en fonction d'une seule condition : la propriété de dérivabilité doit être récursivement représentable. On épure ainsi le concept d'*indécidabilité mathématique* de manière à déterminer les limites concrètes des formalismes. Conséquemment, dans le cas d'une logique ordinale, la propriété de dérivabilité n'est plus vérifiable de façon effective et n'est donc plus récursivement représentable. Au même titre que l'oracle, l'intuition permettrait de déterminer si une *fbf* est une formule ordinale et, par conséquent, d'obtenir la complétude pour les théorèmes *N.T.*

4.6 Complétude et invariance

Turing obtint un résultat partiel de complétude pour les propositions Π_1 : pour toute progression récursivement consistante et toute proposition Π_1 vraie ϕ, il existe un élément a de O où $|a| = \omega + 1$ tel que ϕ est démontrable dans L_a. On obtient bien entendu la complétude pour Π_1, mais cela est tout à fait normal dans la mesure où on atteint les ordinaux transfinis. On combat ainsi le feu par le feu : on utilise un processus *constructif* infini pour annuler l'effet de la diagonalisation. Autrement dit, les axiomes de consistance,

ajoutés les uns aux autres pour former des progressions récursivement consistantes, ne sont en aucun cas utilisés. La seule raison qui explique que ϕ soit démontrable, une fois L_ω atteint, est qu'on obtient au stade ω une définition non-standard des axiomes de L_a. Sans contredit, cela repose sur des définitions imprédicatives et est complètement inutile à des fins de calculabilité. Par ailleurs, ce résultat est dit *partiel* au moins en ce sens que Turing espérait obtenir la complétude pour les problèmes *N.T.* (Π_2). Il écrit :

> ≪I cannot at present give a proof of this (i.e. for *N.T.* problems), but I can give a proof that it is complete as regards a simpler type of theorems than the number-theoretic theorems, viz. those of the form $\Theta(x)$ vanishes identically where $\Theta(x)$ is primitive recursive.≫. (Turing [1939])

Si Turing persiste à croire qu'il est possible d'obtenir la complétude pour les théorèmes *N.T*, c'est parce que ceux-ci sont théoriquement solubles par une machine de Turing avec oracle. Du fait que la bande supplémentaire de l'oracle permet de résoudre tout problème *N.T* particulier, on peut s'attendre à ce qu'une logique ordinale soit complète pour cette classe. Autrement dit, le nombre réel sur la bande peut être associé à la suite ordonnée des formules logiques qui composent une logique ordinale. Nonobstant le fait qu'il est impossible de déterminer par un algorithme si une *fbf* est un λ-terme, on peut donc croire théoriquement possible d'obtenir la complétude pour ces problèmes. Il existe, cependant, un nombre infini de problèmes Π_2 indécidables. Toutefois, pour obtenir la complétude, il serait nécessaire d'avoir un nombre infini de machines de Turing avec oracle, du fait que chaque machine résout bel et bien un seul problème. On sort

ainsi du domaine de la dénombrabilité (i.e. de l'énumération) du fait qu'une machine de Turing avec oracle peut contenir un nombre infini sur sa bande. Par conséquent, on ne peut pas obtenir, par le biais d'une logique ordinale, la complétude pour Π_2.

Feferman montra plus tard, à cet gard, que cela était bel et bien impossible (Feferman [1962]). Le résultat de Turing est aussi dit partiel en fonction de l'inutilité de celui-ci, dans la mesure où le problème qui consiste à reconnaître si une formule est une formule ordinale est déjà plus compliqué que n'importe quel autre problème *N.T.* Kleene écrit à ce sujet :

> ≪[...] its use would afford no theoretic gain, since the recognition that the number which plays the role of ordinal representative in a proof of the logic is actually such comes to the same as the direct recognition of the truth of the proposition proved.≫. (Kleene [1943])

Nous avons montré qu'il est possible de déduire de manière intuitive ce résultat, du fait qu'il s'agit de traiter, dans les deux cas, un ensemble récursivement énumérable. Feferman ajoute :

> ≪Thus the demand on "intuition" in recognizing "which formulae are ordinal formulae" is somewhat greater than Turing suggests. But even in his own terms, there is a failure to test his analysis of purpose against reality. Is it a "spontaneous judgement" without any "conscious train of reasoning" that leads one to recognize a complicated though computable ordering as being

a well-ordering ?≫. (Feferman [1988])

Un autre apport considérable de la recherche de Turing fut bien le résultat concernant l'impossibilité d'obtenir une logique ordinale à la fois complète et invariante. Turing définit la complétude ainsi :

> ≪We say that an ordinal logic Λ is complete if corresponding to each dual formula A there is an ordinal formula Ω_A such that $\Lambda(\Omega_A, A)$ *conv* 2.≫. (Turing [1939])

Il s'agit donc de la complétude pour les théorèmes *N.T.* On pourrait aussi dire que la classe des formules logiques $\Lambda(\Omega)$ est complète si Ω parcourt l'ensemble des formules ordinales puisqu'on traiterait alors aussi toutes les propositions *N.T.* vraies. En plus de la complétude, il est souhaitable pour une logique ordinale d'obtenir l'invariance :

> ≪An ordinal logic is said to be invariant up to an ordinal *a* if, whenever Ω, Ω' are ordinal formulae representing the same ordinal less than *a*, the extent of $\Lambda(\Omega)$ is identical with the extent of $\Lambda(\Omega')$. An ordinal logic is invariant if it is invariant up to each ordinal represented by an ordinal formula.≫. (Turing [1939])

La logique Λ_P (*P* pour Peano), par laquelle Turing obtint son résultat partiel de complétude, consiste à joindre successivement des propositions arithmétiques afin de surmonter ainsi, toujours un peu plus à chaque niveau, les effets du théorème d'incomplétude. Celle-ci, étant complète pour Π_1, n'est toutefois pas invariante. Une autre

logique, Λ_H (H pour Hilbert), est elle invariante et, par conséquent, aussi incomplète. C'est un résultat bien décevant pour Turing puisqu'il anéantit la portée pratique des logiques ordinales.

4.7 Échos du principe d'une logique ordinale

Le texte sur la logique ordinale de Turing est un maillon important dans l'histoire de l'informatique théorique du fait qu'il a contribué grandement à son développement. De plus, Turing initia un courant en fondements des mathématiques qui trouva écho chez Feferman (Feferman [1962, 1988, 2006] et Kreisel (Kreisel [1960]). À cet égard, Feferman réussit à obtenir un résultat de complétude pour les propositions Π_2 en utilisant un principe de réflexivité appliqué à des extensions consistantes autonomes [7]. De plus, l'introduction du concept d'*oracle* est, sans doute, la notion qui a eu le plus grand impact théorique du fait qu'elle est à l'origine de nombreuses avancées en calculabilité, suivant les recherches de Post (Post [1994]) , Sacks (Sacks [1966]) et plusieurs autres. Il est intéressant de s'interroger sur la signification théorique du concept d'oracle dans le texte. En effet, Post soutient que le concept d'oracle est introduit en tant que *side-issue*, ou idée marginale, dans la mesure où ce concept n'est traité que dans le chapitre quatre du texte de Turing :

> ≪In his paper on ordinal logics, Turing presents as a side issue a formulation which can immediately be restated as the general formulation of the

[7] Notons qu'une extension n'est pas une progression comme chez Turing.

> recursive reducibility of one problem to another, and proves a result which immediately generalizes to the result that for any recursively given unsolvable problem there is another of higher degree of unsolvability. ≫. (Post [1944])

Toutefois, le concept d'oracle correspond, dans le contexte du formalisme de Church, à l'intuition du mathématicien. Il s'agit, sans contredit, non pas d'un simple ajout ou d'une question secondaire de la part de Turing, mais bien d'un outil théorique original qui permet de juger de l'implication possible de l'intuition en théorie de la calculabilité et dans les mathématiques en général. Le lien à cet égard a déjà été établi. Un autre apport significatif du texte est l'anticipation de la hiérarchie arithmétique et de la calculabilité relative et ce en ce qui concerne non seulement la calculabilité mais aussi la complexité de fonctions calculables. En effet, Turing écrit :

> ≪We might also expect to obtain an interesting classification of number-theoretic theorems according to *depth*. A theorem which required an ordinal α to prove would be deeper than one which could be proved by the use of an ordinal β less than α.≫. (Turing [1939])

Bref, pour toutes ces raisons, *Systems of logic based on ordinals* est un texte séminal de la littérature qu'il est important de considérer d'un point de vue historique, mais aussi philosophique et mathématique.

CHAPITRE 5

CONCLUSION

5.1 Considérations sur les fondements des mathématiques

L'intuition joue un rôle prépondérant dans la vie (finie) des mathématiciens. Je ne fais pas référence à l'intuition comme s'il s'agissait d'une *faculté* à saisir une soi-disant réalité mathématique. Je parle de l'intuition comme de la capacité du mathématicien à prendre des raccourcis, à faire des choix, en fonction des possibilités *infinies* qui s'offrent à lui. À titre d'exemple, il existe des *assistants* de preuve, à savoir des programmes informatiques, qui aident le mathématicien dans sa démarche démonstrative. Un assistant de preuve idéal consisterait à séparer *exactement*, de manière mutuellement exclusive, la partie formelle des mathématiques de sa partie intuitive. Les assistants de preuves courants (e.g. *Coq*, *Isabelle*) se basent sur le formalisme du λ-calcul du fait, bien entendu, de sa capacité d'abstraction. À cet égard, on peut dire que le projet de Turing consistait à *créer* un assistant de preuve. Un ordinateur ne sait que faire d'un nombre infini : il calcule aussi longtemps que cela lui est possible. La théorie des ensembles transfinis de Cantor introduit un nombre considérable de notions imprédicatives. Une notion est non-prédicative si elle définit un individu appartenant à un certain ensemble en faisant intervenir cet ensemble lui-même. Cantor avait certes beaucoup d'intuition mais le fait de définir imprédicavement la suite des nombres naturels, de vouloir outre-

passer l'infranchissable (ou l'incalculable), semble avoir peu d'intérêt pour le travail du mathématicien qui consiste d'abord à calculer. À cet égard, le paradis cantorien semble être plutôt l'enfer de Hilbert, du moins dans la mesure où c'est en fonction de celui-ci que son projet échoua, et non pas en fonction du théorème d'incomplétude. Le résultat de Gödel n'est qu'une démonstration directe des conséquences obtenues par l'utilisation de tels concepts. Le phénomène d'incomplétude s'établit par le processus non constructif de la diagonalisation à la Cantor en fonction du postulat d'induction de Peano. Mais si, aux premiers abords, l'intuition semble être le *moyen* de combler les trous creusés par l'incomplétude – comme dans le cas des assistants de preuve ou de la logique ordinale de Turing – tel n'est pas vraiment le cas. Où est-il nécessaire de *sauter* par-dessus une valeur non définie (cas de l'opérateur μ) ? Quand sait-on qu'une machine de Turing s'arrêtera ou plutôt, qu'elle bouclera (cas du problème de l'arrêt) ? Comment sait-on qu'un λ-terme peut se convertir à un autre (cas du théorème de Church) ? Il s'agit bel et bien là de problèmes indécidables. Mais tous ces problèmes consistent à trouver une méthode effective, un algorithme, permettant de résoudre d'une manière générale un ensemble infini de problèmes. Il y a une infinité de fonctions partielles. Il y a une infinité de machines de Turing. Il y a une infinité de λ-termes. Il faut, par ailleurs, traiter chaque problème individuellement. Il faut remplacer le quantificateur universel par le quantificateur effini (Gauthier [2002], [2010]). Évidemment, un mathématicien n'aura pas assez de sa vie pour épuiser les ressources illimités des mathématiques. Où est donc le problème ? Syropoulos écrit ceci dans un ouvrage intitulé *Hypercomputation : Computing Beyond*

the Church-Turing Barrier au sujet de la construction de machines de Turing infinies (en théorie, ces machines peuvent résoudre tous les problèmes) :

> The possibility of creating such machines depends mainly on the properties of our universe, For instance, if time and space are continous, in spite of quantum gravity expecting them to be granular, then in principle, it is possible to perform a supertask. Similarly, in a Newtonian universe space, time, and matter are continuous, and thus it is possible to build such machines. (Syropoulos [2008])

S'il y a un seul problème au niveau des fondements mathématiques, c'est le fait que notre imagination peut dominer notre raison. L'argument diagonal est un outil mathématique non constructif. Si nous éradiquons cet instrument – et en même temps tous les autres éléments non constructifs des mathématiques–, nous obtiendrons une théorie mathématique certes plus faible et incomplète, mais une théorie mathématique à l'image de la nature humaine, à l'image de son créateur.

BIBLIOGRAPHIE

[1] Wilhelm Ackermann. Zum hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99:118–133, 1928.

[2] Peter Aczel. Frege structure and the notions of proposition, truth and set (1978). Dans Barwise, éditeur, *The Kleene Symposium*, volume 101, pages 31–59, Juin 1980.

[3] H. P. Barendregt. *The Lambda Calculus : Its Syntax and Semantics*. North-Holland Publishing Co., 1981.

[4] Alessandro Baricco. *Novecento : un monologo*. Feltrinelli, 1994.

[5] Georg Cantor. Über eine Eigenschaft des Inbegriffes aller reelen algebraischen Zahlen. *Journal de Crelle*, 77:258–262, 1874.

[6] Alonzo Church. A set of postulates for the foundation of logic. 33(2):pp. 346–366, 1932.

[7] Alonzo Church. An unsolvable problem of elementary number theory. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.* Dover Publications Inc., 1936.

[8] B. Jack Copeland, éditeur. *The Essential Turing : The ideas that gave birth to the computer age*. Clarendon Press, 2004.

[9] Martin Davis, éditeur. *The undecidable*. Dover Publications Inc., Mineola, NY, 2004. Basic papers on undecidable propositions, unsolvable problems and computable functions, Corrected reprint of the 1965 original [Raven Press, Hewlett, NY ; MR0189996].

[10] John W. Dawson, Jr. The reception of Gödel's incompleteness theorems. Dans *Perspectives on the history of mathematical logic*, pages 84–100. Birkhäuser Boston, Boston, MA, 1991.

[11] Richard Dedekind. *Was sind und was sollen die Zahlen ?* Vieweg v. John, Braunschweig (1965), 7e ed. édition, 1888.

[12] Gilles Deleuze et Félix Guattari. *Qu'est-ce que la philosophie ?* Les Éditions de Minuit, 1991.

[13] Alonzo Church et S. C. Kleene. Formal definitions in the theory of ordinal numbers. *Fund. Math.*, vol. 28:11–21, 1936.

[14] Solomon Feferman. Transfinite recursive progressions of axiomatic theories. *The Journal of Symbolic Logic*, 27(3):pp. 259–316, 1962.

[15] Solomon Feferman. Turing in the land of o(z). Dans *The universal Turing machine : a half-century survey*. Oxford University Press, 1988.

[16] Solomon Feferman. The impact of the incompleteness theorems on mathematics. 53:434–439, 2006.

[17] Solomon Feferman. Gödel's incompleteness theorems, free will, and mathematical thought. Dans Richard Swinburne, éditeur, *Free Will and Modern Science*. Oxford University Press, 2011.

[18] Torkel Franzén. *Inexhaustibility*, volume 16 de *Lecture Notes in Logic*. Association for Symbolic Logic, Urbana, IL, 2004.

[19] Torkel Franzén. *Gödel's theorem : An incomplete guide to its use and abuse*. A K Peters Ltd., Wellesley, MA, 2005.

[20] Yvon Gauthier. *Internal logic. Foundations of mathematics from Kronecker to Hilbert*. Dordrecht, Kluwer "Synthese Library", vol. 310, 2002.

[21] Yvon Gauthier. *Logique Arithmétique. L'arithmétisation de la logique*. Collection « Logique de la science », PUL, Québec, 2010.

[22] Gerhard Gentzen. Die Widerspruchsfreiheitsbeweis der reinen Zahlentheorie. *Mathematische Annalen*, (112):93–565, 1936.

[23] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls*. Thèse de doctorat, Université de Vienne, 1929.

[24] Kurt Gödel. On formally undecidable propositions of P*rincipia* M*athematica* and related systems. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 4–38. Dover Publications Inc., 1931.

[25] Kurt Gödel. *Kurt Gödel : collected works. Vol. V*. The Clarendon Press Oxford University Press, Oxford, 2003. Correspondence H–Z, Edited by Solomon Feferman, John W. Dawson, Jr., Warren Goldfarb, Charles Parsons and Wilfried Sieg.

[26] D. Hilbert et W. Ackermann. *Grundzüge der theoretischen Logik*. Berlin, 1931. Sechste Auflage, Die Grundlehren der mathematischen Wissenschaften, Band 27.

[27] David Hilbert. *Grundlagen der Geometrie*. B.G. Teubner, 1ière éd. édition, 1899.

[28] David Hilbert. Mathematical problems. *Bull. Amer. Math. Soc. 8*, pages 437–479, 1902.

[29] David Hilbert. The new grounding of mathematics. first report. Dans P. Mancosu, éditeur, *From Hilbert to Brouwer. The Debate on the Foundations of Mathematics in the 1920s*. Oxford University Press, 1922.

[30] David Hilbert. Über das Unendliche. *Math. Ann.*, 95(1):161–190, 1926.

[31] David. Hilbert. Naturerkennen und logik. Dans *Gedenkband*, September 1930.

[32] David Hilbert. Probleme der Grundlegung der Mathematik. *Math. Ann.*, 102(1): 1–9, 1930.

[33] S. C. Kleene. General recursive functions of natural numbers. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, numbéro 1, pages 236–252. Dover Publications Inc., 1936.

[34] S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3: 150–155, 1938.

[35] S. C. Kleene. Recursive predicates and quantifiers. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 254–287. Dover Publications Inc., 1943.

[36] S. C. Kleene et J. B. Rosser. The inconsistency of certain formal logics. *Ann. of Math. (2)*, 36(3):630–636, 1935.

[37] G. Kreisel. Ordinal logics and the characterization of informal concepts of proof. Dans *Proc. Internat. Congress Math. 1958*, pages 289–299. Cambridge Univ. Press, New York, 1960.

[38] Jean Ladrière. *Les limitations internes des formalismes. Etude sur la signification du théorème de Gödel et des théorèmes apparentés dans la théorie des fondements des mathématiques*. Collection de Logique Mathematique, Serie B, II. E. Nauwelaerts, Louvain, 1957.

[39] Gottfried Wilhelm Leibniz. *Essais de théodicée*. Flammarion (1990), 1710.

[40] Friedrich Nietzsche. *L'Antéchrist*. Gallimard, 1895.

[41] Giuseppe Peano. Arithmetices principia, nova methodo exposita. 1889.

[42] Roger Penrose. *The emperor's new mind*. The Clarendon Press Oxford University

Press, New York, 1989. Concerning computers, minds, and the laws of physics, With a foreword by Martin Gardner.

[43] Roger Penrose. *Shadows of the mind*. Oxford University Press, Oxford, 1994. A search for the missing science of consciousness.

[44] Emil Post. Recursively enumerable sets of positive integers and their decision problems. Dans *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 304–337. Dover Publications Inc., 1944.

[45] Emil L. Post. *Solvability, provability, definability : the collected works of Emil L. Post*. Contemporary Mathematicians. Birkhäuser Boston Inc., Boston, MA, 1994. Edited and with an introduction by Martin Davis.

[46] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):pp. 358–366, 1953.

[47] J. B. Rosser. Extensions of some theorems of Gödel and Church. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 231–235. Dover Publications Inc., 1936.

[48] J. B. Rosser. An informal exposition of proofs of Gödel's theorem and Church's theorem. Dans *The Undecidable : Basic Papers on Undecidable Proposition, Un-*

solvable Problems and Computable Functions., pages 223–229. Dover Publications Inc., 1939.

[49] Gerald E. Sacks. *Degrees of unsolvability*. Princeton University Press, 1966.

[50] Michael Sipser. *Introduction to the theory of computation*. Thomson, Inc., San Diego, CA, 2006.

[51] Theodore A. Slaman. Recursion theory in set theory. Dans *Computability theory and its applications (Boulder, CO, 1999)*, volume 257 de *Contemp. Math.*, pages 273–278. Amer. Math. Soc., Providence, RI, 2000.

[52] Raymond Smullyan. Gödel's incompleteness theorems. Dans *The Blackwell guide to philosophical logic*, Blackwell Philos. Guides, pages 72–89. Blackwell, Oxford, 2001.

[53] Gabriel Sudan. Sur le nombre transfini ω^{ω}. *Bulletin Math. Soc. Roumaine des sciences*, 30:11–30, 1927.

[54] Apostolos Syropoulos. *Hypercomputation*. Springer, New York, 2008. Computing beyond the Church-Turing barrier.

[55] René Thom. *Paraboles et Catastrophes*. Flammarion, 1983.

[56] Alan M. Turing. Computability and λ-definability. *The Journal of Symbolic Logic*, (2):153–163., 1936.

[57] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 115–153. Dover Publications Inc., 1937.

[58] Alan M. Turing. Systems of logic based on ordinals. Dans Martin Davis, éditeur, *The Undecidable : Basic Papers on Undecidable Proposition, Unsolvable Problems and Computable Functions.*, pages 154–222. Dover Publications Inc., 1939. Thesis (Ph.D.)–Princeton University.

[59] John von Neumann. *Collected works. Vol. V : Design of computers, theory of automata and numerical analysis*. General editor : A. H. Taub. A Pergamon Press Book. The Macmillan Co., New York, 1963.

[60] Alfred North Whitehead et Bertrand Russell. *Principia Mathematica (3 vols)*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1928.