L'hyper-calcul : de la logique a la physique

Florent Franchette

Université Paris 1 Panthéon-Sorbonne
Ecole doctorale de Philosophie
IHPST

Thèse

pour l'obtention du grade de docteur en Philosophie
de l'Université de Paris 1 Panthéon-Sorbonne

Soutenance prévue le jour mois année

Florent FRANCHETTE

L'HYPER-CALCUL
De la logique à la physique

Sous la direction de Jacques DUBUCS et d'Anouk BARBEROUSSE

Composition du Jury :

Anouk BARBEROUSSE	Professeure à l'Université de Lille 1
Jack COPELAND	Professeur à l'Université de Canterbury
Jacques DUBUCS	Directeur de recherches au CNRS
Gualtiero PICCININI	Maître de conférences à l'Université du Missouri, St. Louis
Hervé ZWIRN	Directeur de recherches au CNRS

Année : 2013

Florent FRANCHETTE

L'HYPER-CALCUL
De la logique à la physique

Thèse

pour l'obtention du grade de docteur en Philosophie
de l'Université de Paris 1 Panthéon-Sorbonne

Sous la direction de Jacques DUBUCS et d'Anouk BARBEROUSSE

A Marie-Thérèse et Jacques,

Table des matières

Remerciements

Ce travail est le résultat de trois années de recherche sous la direction de Jacques Dubucs et d'Anouk Barberousse dont les rôles furent décisifs pour cette thèse.

Il serait en fait plus juste de dire que leur rôle respectif dépasse le cadre de cette thèse car ils ont contribué, chacun à leur manière, à ma formation philosophique. Jacques Dubucs m'a tout d'abord enseigné les deux versants de la logique, à savoir la logique formelle et la philosophie de la logique, en préservant toujours le difficile équilibre entre formalisme logique et réflexion philosophique. Anouk Barberousse m'a quant à elle initié à la philosophie générale des sciences tout en m'apprenant que la rigueur scientifique faisait partie de l'exigence philosophique.

Mais c'est pendant l'écriture de cette thèse que leurs soutiens furent aussi irremplaçables que précieux. Leurs conseils, leur disponibilité et leurs encouragements sont tout ce dont le doctorant a tant besoin. J'ai énormément appris à leurs côtés car ils ont été pour moi des modèles à la fois comme professeurs, pédagogues et chercheurs. Je leur en suis infiniment reconnaissant.

Je remercie profondément Jack Copeland, Gualtiero Piccinini et Hervé Zwirn d'avoir accepté de faire partie de mon jury de thèse.

A bien des égards, ce travail n'aurait pas été possible sans l'aide de Jack Copeland et de Diane Proudfoot. En tant que scientifiques, ils ont été les premiers à étudier l'hyper-calcul et les thèses que je défends dans ce travail ne sont pas étrangères aux leurs. En tant qu'amis, Jack et Diane m'ont accueilli à l'Université de Canterbury et ont accepté de faire le voyage depuis la Nouvelle-Zélande pour assister à ma soutenance : qu'ils soient ici remerciés pour leur disponibilité et leur gentillesse.

Depuis son poste de directeur de l'école doctorale de philosophie à celui de directeur de l'IHPST, Jean Gayon m'a toujours fait confiance et a été plus d'une fois à mon écoute. Je souhaite lui adresser ma profonde gratitude.

Je tiens à remercier chaleureusement l'Université Paris 1 et tout particulièrement Chantal Jaquet, directrice de l'école doctorale de philosophie, pour leurs soutiens constants depuis trois ans. L'école doctorale, en plus de m'avoir fait confiance en m'attribuant un contrat doctoral, m'a ainsi permis de participer à de nombreuses conférences à l'étranger.

Cette thèse doit aussi beaucoup aux nombreuses discussions que j'ai entretenues lors de conférences, de cours et bien sûr de déjeuners. Je souhaite tout particulièrement remercier Susana Berestovoy, Olivier Bournez, Cristian Calude, José-Félix Costa, Gilles Dowek, Alexei Grinbaum, Jean-Baptiste Joinet, Cyrille Imbert, Jean Mosconi, Marcin Mostowski, John Norton, Carlo Proietti, Paula Quinon, François Rivenc, Philippe de Rouilhan, Oron Shagrir, Mark Sprevak, Mike Stannett, Karl Svozil, Raymond Turner, Marion Vorms, Pierre Wagner et Hector Zénil.

Durant ces trois années, j'ai choisi de travailler principalement à l'IHPST. Jean Gayon, directeur de l'institut, ainsi que toute l'équipe administrative comprenant Alexandre Roulois, Peggy Tessier, Armelle Thomas et Sandrine Souraya ne sont pas étrangers à ce choix. Je remercie toute leur bonne humeur et l'aide qu'ils m'ont souvent apportée.

Il serait malhonnête de ne pas témoigner ma gratitude aux nombreux doctorants et amis qui m'ont aidé et soutenu depuis le tout début : Matthieu Amat, Vincent Ardourel, Smaïl Bouaziz, Julien Boyer, Thomas Boyer, Stéphane Copreaux, Marie Darrason, Romain Gransire, Vincent Israël-Jost, Julie Jebeile, Laurent Jodoin, Gladys Kostyrka, Lucy Laplane, Gauvain Leconte, Jérôme Lescane, Johannes Martens, Alberto Naibo, Maël Pegny, Gaëlle Pontarotti, Hélène Richard et Aurélien Tonneau.

Enfin, ce travail ne serait rien sans le soutien indéfectible de ma famille. Je rends ainsi hommage à Claire Baudier, Emmanuel Bézart, Eric Franchette, Amélie Franchette, Sophie Franchette, Jean Kali, Elyette Nautrel, Julie Naturel et Jacques Naturel.

Avertissements

Si ce travail est avant tout une réflexion philosophique, il contient néanmoins de nombreux passages formels. Ces passages n'ont pas fonction de se substituer à l'argumentation mais bien de la renforcer.

J'ai tenté autant que faire se peut de rendre autonome le contenu de ce travail en définissant tous les concepts utilisés. Les concepts les plus importants sont en outre détaillés dans les annexes.

Enfin, un index est à la disposition du lecteur.

Introduction générale

> So, hype or computation ? At this juncture, it seems the jury is still out, but the trial promises to be riveting.
>
> Christof Teuscher and Moshe Sipper

Ce travail se situe à la frontière entre philosophie, logique et informatique. Mon but est de défendre une thèse philosophique sur les ordinateurs et les limites de ce qui est calculable. Contrairement à une position largement répandue, je souhaite défendre la thèse selon laquelle les ordinateurs sont des outils indispensables pour établir les limites de ce qui est calculable. Plus précisément, je pense que pour établir ces limites, il est indispensable de faire appel aux calculs réellement effectués par les ordinateurs.

Dans cette introduction, je commence par présenter le problème traité dans ce travail. Cette présentation débute par une première exposition du problème et se poursuit par une explication détaillée des points importants préalablement introduits. Je termine par énoncer le plan de l'argumentation.

1 Première approche du problème

On considère généralement les ordinateurs comme des machines indispensables à l'exécution de nombreuses tâches. Les ordinateurs sont considérés comme tel car sans eux l'exécution de ces tâches serait impossible. Pour

illustrer cette position communément admise, citons deux exemples de tâches pour lesquels les ordinateurs sont devenus indispensables [1] :

1. **Mémoriser des données.** De même qu'il est apparu nécessaire de garder des traces de tous les textes imprimés - l'une des raisons d'être de la bibliothèque nationale de France ou d'autres bibliothèques équivalentes dans le monde - il est nécessaire à l'heure actuelle de garder des traces des textes rédigés sur internet. Le problème est bien sûr d'une tout autre nature car certaines pages web sont modifiées chaque jour, voir chaque heure ou chaque minute [Delahaye, 2006].

   Plusieurs projets dans le monde se sont attaqués au problème. Le plus avancé est le projet américain *The Internet Archive* (www.archive.org) fondé en 1996. Les données sont accessibles au public grâce au système dénommé *The Wayback Machine* qui contient plus de 5 péta-octets de données - 5.10^{15} octets. A titre de comparaison, l'enregistrement d'un film en très bonne qualité est estimé à 3 giga-octets - 3.10^{9} octets ; *the Wayback Machine* peut donc actuellement enregistrer environ 2 millions de films.

2. **Effectuer des calculs.** L'ordinateur *Sequoia* d'IBM peut exécuter 16.10^{15} instructions par seconde. Cette puissance de calcul permet en particulier de calculer les 2 700 milliards premières décimales de π. Toujours par souci de comparaison, *Deep Blue* - l'ordinateur qui a battu Kasparov aux échecs - était 3000 fois moins puissant que *Sequoia*.

Ces deux exemples permettent d'identifier précisément la relation qui existe entre les ordinateurs et les êtres humains : les ordinateurs repoussent les limites de l'être humain en lui fournissant un espace mémoire et une vitesse qui lui permet d'exécuter des tâches impossibles à réaliser en pratique sans l'aide des ordinateurs. Dit autrement, les ordinateurs déterminent ce que l'être humain peut mémoriser, ce qu'il peut calculer, et chaque nouvelle avancée technologique, chaque ordinateur nouvellement construit élargie ses

1. Ces deux exemples ne sont pas exhaustifs. Les ordinateurs sont aussi indispensables pour simuler les systèmes physiques tels que les climats [Braconnot and Marti, 2002] ou pour exécuter des tâches industrielles - soudage, peinture, assemblage.

possibilités [Humphreys, 2004]. En bref, les ordinateurs définissent les limites pratiques de ce qui est calculable.

Cependant, réalisons l'expérience de pensée suivante qui a été proposée en 1936 par Turing [Turing, 1936]. Imaginons un mathématicien en train d'exécuter un calcul à la main - par exemple le calcul des décimales de π - et supposons (1) qu'il n'est pas limité en fournitures - papier, feutres ; (2) qu'il n'est pas affecté par la fatique ou par tout autre problème de santé ; et (3) qu'il dispose d'un temps arbitrairement grand mais fini. Autrement dit, si le mathématicien exécute ses calculs sans l'aide d'aucune machines - ordinateurs, machines à calculer, *etc.* - il n'est en contrepartie plus restreint à une quantité de ressources physiques donnée, cette dernière pouvant être arbitrairement grande.

A présent, posons la question suivante : en admettant les hypothèses précédentes, est-ce que les ordinateurs définissent ce qui peut être calculé par un être humain ? Les travaux de Turing ont permis de répondre négativement à cette question : tout ce que peut calculer un ordinateur - quels que soient ses capacités - peut être calculé par un mathématicien qui n'est plus limité par le temps ou par l'espace dont il dispose [2]. De ce point de vue, les capacités des ordinateurs n'ont aucune influence sur ce que peut mémoriser ou calculer l'être humain ; ils ne sont plus requis pour élargir ses possibilités.

Cette dernière affirmation semble pourtant reposer sur un artifice puisque la notion de limite utilisée par Turing n'est pas celle qui est communément admise. La limite dont fait référence Turing est en effet définie *en principe*, c'est-à-dire en faisant abstraction des ressources physiques utilisées lors des calculs. Par exemple, elle ne dépend pas du temps ou de l'espace mémoire dont dispose le calculateur. Cette limite est donc en contradiction avec l'expérience quotidienne qui montre que la limite entre ce qui est calculable et ce qui ne l'est pas est définie *en pratique*, à savoir en prenant en compte la quantité de ressources que possède le calculateur.

2. Il est à préciser que Turing n'est pas l'auteur de cette affirmation puisque ses travaux ne sont pas liés directement aux ordinateurs - *cf.* chapitre 1. Néanmoins, les travaux de Turing ont permis d'établir cette affirmation.

Pourquoi les limites du calculable devraient être définies en principe alors que l'expérience quotidienne montre que ces limites dépendent des capacités - vitesse, espace mémoire - des êtres humains ou des ordinateurs ? La réponse de Turing est la suivante : si on considère les limites de ce que l'on peut calculer en fonction des capacités du calculateur, comment être sûr que ce qui n'est pas calculable à un moment donné ne se révèlera pas calculable par la suite ? La notion de limite qui serait ainsi obtenue resterait relative à une époque donnée, puisque, pour pouvoir définir une telle limite il serait requis de faire référence aux avancées technologiques qui sont en constante évolution. S'affranchir du support technologique - et en particulier des capacités des ordinateurs - devient alors une nécessité si on souhaite définir les limites de ce qui est calculable.

Une analyse des limites du calcul qui est indépendante des facteurs technologiques a été entreprise par les logiciens des années 1930. En particulier, Church et Turing ont défini des outils formels permettant d'établir que certaines fonctions mathématiques ne sont pas calculables [Church, 1936], [Turing, 1936]. Ces fonctions ne sont pas calculables au sens où il est impossible de calculer l'ensemble de leurs valeurs, et cela indépendamment de la puissance des ordinateurs. Dit autrement, on ne serait pas capable de calculer de telles fonctions même si l'on disposait de ressources inépuisables.

Les résultats de Church et Turing renforcent ainsi la thèse selon laquelle les limites du calcul ne sont pas définies relativement à un niveau technologique particulier :

> Il me semble que l'on a pour la première fois réussi à proposer une définition absolue d'une intéressante notion épistémologique, c'est-à-dire, d'une notion qui ne dépend pas du formalisme choisi [Gödel, 1946, p. 84].

D'après Gödel, la définition proposée par Church et Turing est à la fois épistémologique et absolue car elle permet d'acquérir une connaissance, celle de savoir si une fonction est calculable, qui ne dépend ni de l'outil formel que l'on choisit ni du niveau technologique que l'on possède. Autrement dit, la conception d'outils logiques ou la construction d'ordinateurs n'apporterait rien de nouveau à propos des limites du calcul établies par les logiciens lors

de la première moitié du XX^{e} siècle.

Considérer les limites du calcul établies par Turing et Church comme des limites *absolues* - au sens de Gödel - revient selon moi à soutenir une thèse philosophique forte sur les ordinateurs et les limites du calcul. Cette thèse, qui est implicitement admise depuis la seconde moitié du XX^{e} siècle, peut être énoncée de la manière suivante : même si les ordinateurs sont en pratique indispensables pour calculer ou mémoriser des données, ils sont en principe négligeables lorsque l'on cherche à définir les limites de ce qui est calculable. Les ordinateurs sont négligeables car leurs capacités calculatoires - vitesse de calcul, espace de stockage, *etc.* - n'ont aucune influence sur les limites définies par Turing et Church.

Cette dernière thèse implique en outre que les limites du calcul ne doivent plus faire référence aux types de calculateurs - humains ou machines - traditionnellement différenciés par leurs capacités. Plus précisément, ces limites peuvent être définies sans les reconduire à des distinctions liées aux calculateurs - par exemple, une distinction entre ce qui est calculable par l'être humain et ce qui est calculable par l'ordinateur. L'étude du calculateur - qu'il soit humain ou machine - n'est donc plus pertinente pour étudier les limites du calcul.

Dans ce travail, je souhaite cependant m'opposer à la thèse selon laquelle la construction d'ordinateurs ne pourrait pas modifier les limites du calcul qui ont été proposées. Ma position est qu'il est indispensable d'étudier les ordinateurs pour établir les limites du calculable, que ces limites soient définies en principe ou en pratique. En ce sens, je remets en cause la distinction en principe / en pratique sur laquelle se fondent les logiciens des années 1930 pour établir des limites indépendantes des avancées technologiques. Dit autrement, je pense que ces limites ne sont pas absolues au sens où certains ordinateurs différents des ordinateurs actuels sont capables de les franchir.

Pour être plus précis, ma stratégie consiste à analyser la validité de *l'hyper-calcul*, terme introduit par Copeland et Proudfoot qui dénote la possibilité de calculer au-delà des limites établies par Church et Turing [Copeland and Proudfoot, 1999]. En particulier, mon objectif est d'évaluer deux thèses sur l'hyper-calcul :

1. Il est logiquement possible d'hyper-calculer.
2. Il est physiquement possible d'hyper-calculer.

Dans le cas où elles s'avéreraient vraies, ces deux thèses remettraient en cause les limites préalablement établies par Turing, limites que certains nomment *barrière de Turing* [Delahaye, 2006]. La première thèse signifie en effet que la découverte de nouveaux outils logiques pourrait modifier les limites du calculable. La seconde thèse implique quant à elle qu'il serait possible de franchir la barrière de Turing en construisant physiquement certains dispositifs capables d'hyper-calculer. La vérité de ces deux thèses modifierait ainsi en profondeur les limites du calcul définies dans les années 1930.

Afin de déterminer si l'étude des ordinateurs est indispensable pour définir ce qui est calculable, il devient indispensable suite aux travaux de Copeland & Proudfoot d'évaluer la possibilité logique et physique de l'hyper-calcul. Pour mieux comprendre cette dernière affirmation, je détaille dans la suite de cette introduction chacun des points importants qui ont été introduits jusqu'ici. Plus précisément, je commence par expliquer pourquoi les logiciens tels que Turing et Church ont privilégié une approche en principe pour déterminer les limites du calcul. Je présente ensuite l'hyper-calcul, de ses origines conceptuelles aux problèmes contemporains qu'il soulève. Je termine cette introduction par énoncer le mouvement général de l'argumentation.

2 Des limites en pratique du calcul à ses limites en principe

Même si les limites du calcul peuvent être définies de plusieurs manières [3], les limites que pourrait modifier l'hyper-calcul ont pour principale particularité d'être définies *en principe*, c'est-à-dire en faisant abstraction des ressources physiques utilisées lors des calculs. Par exemple, les limites en principe ne dépendent pas du temps ou de l'espace mémoire dont dispose le calculateur. Cette section a ainsi pour objectif d'expliquer les raisons qui ont

3. Consulter en particulier la thèse de Imbert sur *la complexité intrinsèque* de la nature [Imbert, 2008].

poussé les logiciens tels que Turing et Church à privilégier une approche en principe pour déterminer les limites du calcul [4].

2.1 La nature relative des limites en pratique

Pourquoi est-il impossible d'effectuer certains calculs ? Si cette impossibilité peut avoir des origines diverses, elle provient généralement d'un manque de ressources physiques. Un exemple très simple permet de comprendre cette affirmation. Supposons que l'on souhaite calculer les décimales de π à l'aide de la *formule de Leibniz*

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Le problème est que cette série converge si lentement qu'il est nécessaire de calculer ses 300 premiers termes pour obtenir deux décimales exactes. Pire encore, le calcul des cent premières décimales demande plus de termes qu'il n'y a de particules dans l'univers. La réalisation d'un tel calcul nécessite par conséquent une quantité de ressources - temps, espace de calcul - que l'on ne possède pas ; c'est donc à cause d'un manque de ressources physiques que l'on ne peut pas effectuer ce calcul.

Toutefois, un calcul qui est infaisable en pratique à un moment donné peut ne plus l'être à un autre moment. Il existe en effet deux raisons à cela :

1. **Le calcul en pratique dépend des avancées technologiques**. Puisqu'un calcul est faisable en pratique si l'on dispose des ressources nécessaires pour l'exécuter, il est clair que la faisabilité des calculs est proportionnelle à l'augmentation de la puissance du calculateur et en particulier de celle des ordinateurs, machines qui se sont substituées aux calculateurs humains au cours du temps. Ainsi l'ABC (*Atanasoff-Berry Computer*) de 1942 ne peut pas effectuer autant de calculs que Sequoia, ordinateur fabriqué en 2012 par IBM et pouvant exécuter 16×10^{15} instructions par seconde.

4. Pour faciliter la lecture, j'utilise à partir de maintenant les expressions *limites en principe* et *limites pratiques* à la place des expressions *limites en principe du calcul* et *limites en pratique du calcul*.

2. **Le calcul en pratique dépend des outils formels**. Cela signifie que les calculs infaisables en pratique peuvent parfois être réalisés une fois que certains outils formels ont été conçus. Plus précisément, la découverte de formules ou *algorithmes* plus efficaces que celle de Leibniz - à savoir d'algorithmes qui nécessitent moins d'étapes de calcul pour obtenir une décimale - a permis de calculer les cent premières décimales de π. Conjointement aux avancées technologiques, la découverte de nouveaux algorithmes a en particulier permis de calculer sa 2.5^{11}-ième décimale.

Le fait que les limites pratiques dépendent des avancées technologiques et de la découverte de nouveaux outils mathématiques soulève immédiatement la question suivante : peut-on encore parler de *limites* pratiques du calcul ? Autrement dit, est-ce que la construction d'ordinateurs toujours plus puissants et la découverte d'algorithmes toujours plus efficaces ne sont-elles pas des preuves qu'il n'existe en réalité aucune limite pratique ? S'il est vrai que les limites pratiques sont constamment repoussées, il existe en fait certains arguments en faveur de la thèse selon laquelle la puissance des ordinateurs et l'efficacité des algorithmes ne peuvent pas être améliorés indéfiniment.

Il existe tout d'abord une limite physique à la puissance des ordinateurs. La puissance des ordinateurs repose en effet sur les améliorations des composants électroniques - appelés *transistors* - et en particulier sur leur miniaturisation : plus le nombre de transistors est important et plus l'ordinateur calcule vite. En 1965, Gordon Moore remarque que depuis l'invention du circuit intégré, le nombre de transistors par puce électronique augmente de manière exponentielle [Moore, 1965]. Plus précisément, Moore prédit que leur nombre doublera tous les 18 mois, ce que l'on dénomme abusivement *la loi de Moore*[5]. Même si cette loi est pour l'instant vérifiée chaque année, la précision

5. La loi de Moore n'est pas une loi mathématique ou physique mais une règle hypothétique qui résulte d'une combinaison de divers facteurs : « les succès des techniques informatiques auprès du public (facteur social) et la concurrence entre firmes (facteur économique) déterminent les bénéfices de l'industrie des semi-conducteurs, qui fixent à leur tour l'investissement en recherche de cette industrie (facteur technologique), lui permettant de surmonter les obstacles qu'oppose la réalité matérielle (facteur physique) » [Delahaye, 2002, p. 79].

nécessaire à la miniaturisation des transistors empêchera la prédiction de Moore de continuer *ad infinitum* et l'évolution de la puissance des ordinateurs devra obligatoirement s'arrêter.

L'existence d'une limite à l'efficacité des algorithmes est plus complexe à établir bien qu'un élément de réponse puisse être apportée par *la théorie de la complexité* [Papadimitriou, 1994]. L'origine de la théorie de la complexité remonte aux années 1960 et provient d'une volonté de comparer l'efficacité des algorithmes. Comparer l'efficacité des algorithmes était en effet difficile car chaque algorithme était soumis aux capacités calculatoires de la machine qui l'exécutait, capacités qui dépendaient du processeur, du langage de programmation ou du compilateur utilisés. Une approche formelle et indépendante des facteurs matériels était donc nécessaire pour évaluer l'efficacité des algorithmes. En particulier, la théorie de la complexité énonce une hypothèse à propos de cette efficacité. Informellement, cette hypothèse signifie qu'il n'existe aucun algorithme pouvant résoudre efficacement certains problèmes [Sipser, 1992]. En d'autres termes, l'hypothèse affirme que le nombre d'étapes de calcul nécessaire pour résoudre ces problèmes n'est pas faisable en pratique, et cela quel que soit l'algorithme que l'on utilise. Si elle est un jour démontrée, cette hypothèse devra donc être considérée comme une limite de ce qui est en pratique calculable.

Même s'il existe des arguments en faveur de limites à la puissance des ordinateurs et à l'efficacité des algorithmes, ces limites ne sont pas absolues au sens où elles pourraient être modifiées en fonction des avancées scientifiques. Par exemple, les recherches actuelles en informatique quantique apportent des raisons de penser que les limites pratiques seront dépassées. D'un point de vue technologique, les scientifiques tentent actuellement de construire des ordinateurs quantiques utilisant une technologie composée de processeurs et de transistors quantiques[6]. D'un point de vue algorithmique, la conception d'algorithmes quantiques a permis de montrer qu'il est possible de résoudre efficacement certains problèmes pour lesquels aucune solution efficace non

6. L'entreprise *D-wave* tente en particulier de commercialiser les premiers ordinateurs quantiques : `http://www.dwavesys.com/en/dw_homepage.html`. Ces ordinateurs quantiques auraient de plus le potentiel de modifier les limites pratiques [Johnson et al., 2011].

quantique n'est connue [Shor, 1994].

Les recherches en informatique quantique soulèvent ainsi un problème intrinsèque aux limites pratiques : comment déterminer si un calcul infaisable à un moment donné ne se révèlera pas faisable par la suite ? Puisque les limites pratiques évoluent parallèlement aux avancées scientifiques, il semble que ces limites ne puissent pas être utilisées comme critère objectif pour différencier ce qui est calculable de ce qui ne l'est pas. Pour trouver un tel critère, il convient donc de poser la question suivante : peut-on établir des limites du calcul qui seraient indépendantes à la fois de la construction d'ordinateurs et de la conception de nouveaux algorithmes ? Autrement dit, existe-t-il une limite non pas en pratique mais en *en principe* entre le calculable et le non calculable ? La réponse à cette question est positive est fut apportée par les logiciens des années 1930.

2.2 La nature absolue des limites en principe

Afin d'analyser les limites du calcul indépendamment des avancées technonologiques et algorithmiques, les logiciens des années 1930 ont dû faire abstraction des ressources physiques utilisées lors des calculs. Cette abstraction permet en particulier d'annihiler l'impact que peut avoir l'augmentation de la puissance des ordinateurs et l'amélioration de l'efficacité des algorithmes. Autrement dit, les ordinateurs ne sont plus différenciés par leurs gains en temps ou en espace de stockage mais uniquement en fonction de leur capacité à calculer une fonction donnée, que le calcul nécessite quelques minutes ou plusieurs centaines d'années.

Une telle abstraction permet ainsi à la notion de calcul de revêtir deux sens différents :

1. Comme je l'ai indiqué dans la section précédente, un calcul est le plus souvent exécuté *en pratique*, c'est-à-dire en utilisant les ressources physiques - temps, espace mémoire - qui sont nécessaires pour mener à bien le calcul.
2. Mais un calcul peut être aussi exécuté *en principe*, c'est-à-dire indépendamment des ressources physiques qui s'avèrent nécessaires pour effec-

tuer le calcul. Cela signifie intuitivement que le calculateur, qu'il soit humain ou machine, ne manque jamais de feuilles, de crayons, de temps, d'espace mémoire, *etc.*

Le travail des logiciens des années 1930 a été de déterminer les calculs pouvant être exécutés en principe. Toutefois, comment définir ce qu'est un calcul pouvant être exécuté en principe alors que les ressources physiques - *mètre-étalon* de ce qui est faisable en pratique - ont été mises de côté ? Pour répondre à cette question, les logiciens ont introduit la notion de *procédure effective* et ont défendu la thèse selon laquelle les calculs exécutables en principe sont les calculs pouvant être exécutés par une procédure effective.

Informellement, une procédure P est *effective* lorsque :

1. P est définie à partir d'un nombre fini d'instructions comportant chacune un nombre fini de symboles ;
2. P produit le résultat du calcul en un nombre fini d'étapes ;
3. P peut être exécutée en principe, à savoir en faisant abstraction des ressources physiques liés au calcul ;
4. P ne demande aucune perspicacité ni ingéniosité pour être exécutée.

Si la notion de procédure effective semble correspondre à l'idée intuitive de calcul, une définition informelle de cette notion n'est toutefois pas suffisante pour évaluer les limites en principe. En effet, il ne suffit pas de tester l'ensemble des procédures effectives connues pour affirmer si un calcul n'est pas exécutable en principe. Si tel était le cas, les limites en principe possèderaient le même défaut que les limites pratiques, à savoir celui d'être dépendant des avancées scientifiques et de l'état de nos connaissances. Pour définir les limites en principe, il est donc nécessaire de formaliser la notion de procédure effective afin de pouvoir établir qu'il n'existera jamais de procédure effective pour un calcul donné.

La formalisation de la notion de procédure effective a été entreprise par les logiciens des années 1930 tels que Turing et Church. Ces derniers ont commencé par remplacer la notion informelle de *calcul exécutable* par la notion formelle de *fonction calculable définie sur les entiers naturels*. Ils ont ensuite

formellement caractérisé les fonctions calculables par procédure effective [7] :

- Les fonctions générées par les systèmes de Herbrand et Gödel [Gödel, 1934].
- Les fonctions λ-définissables [Church, 1936].
- Les fonctions calculables par machine de Turing [Turing, 1936].
- Les fonctions récursives [Kleene, 1936].
- Les langages générés par les systèmes canoniques [Post, 1941].

Bien qu'étant définies de manières différentes, ces formalisations sont équivalentes au sens où elles définissent le même ensemble de fonctions. Cette équivalence est un argument très fort en faveur de la thèse selon laquelle les logiciens des années 1930 ont réussi à formaliser la *calculabilité*, à savoir ce qui peut être calculé en principe :

> Le fait que deux définitions si différentes [les fonctions λ-définissables et les fonctions récursives] de la calculabilité effective se révèlent équivalentes est une raison supplémentaire de penser que ces définitions constituent une caractérisation de cette notion, consistante avec l'idée intuitive que nous en avons [Church, 1936, p.90].

De façon canonique, la thèse selon laquelle la calculabilité a été correctement formalisée par Turing et Church est exprimée dans la littérature par la () :

Thèse (Thèse de Church-Turing)
Les fonctions calculables par procédure effective sont calculables par machine de Turing

Il est important de comprendre que cette thèse faisant le lien entre une notion informelle - celle de procédure effective - et une notion formelle - celle de machine de Turing - n'est pas un théorème mathématique. Toutefois, même s'il n'existe actuellement aucune preuve mathématique de cette thèse, aucun contre-exemple n'a pu être fourni et la grande majorité de la communauté scientifique pense qu'il n'en existe pas - *cf.* chapitre 1. Il est aussi important

7. Les notions de machine de Turing et de fonction récursive sont présentées en détails dans les annexes A et B.

de préciser que la formalisation choisie - ici celle de machine de Turing - n'a pas d'incidence particulière. J'ai choisi la machine de Turing comme formalisation canonique de la calculabilité mais mon choix aurait pu se porter sur la λ-définissabilité ou sur la récursivité. Ce choix n'est cependant pas purement arbitraire car

> Le concept de machine de Turing a pour avantage d'identifier la notion d'effectivité dans un sens intuitif et immédiat, c'est-à-dire sans qu'il soit nécessaire de prouver des théorèmes préliminaires [Church, 1937, p. 42].

En acceptant cette équivalence entre les notions de procédure effective et de machine de Turing, Turing et Church ont prouvé un résultat fondamental à propos des limites en principe du calcul : il existe des fonctions - en réalité une infinité - qui sont hors de portée du calcul par procédure effective. Plus encore, ils ont réussi a exhiber certaines de ces fonctions. La *fonction diophantienne* définie ci-dessous est un exemple simple et percutant de fonction non calculable par procédure effective :

Définition (Fonction diophantienne)
Une équation diophantienne est une équation dont les cœfficients sont des nombres entiers et dont les solutions recherchées sont également entières. Etant donnée une telle équation codée par un entier naturel n, la fonction diophantienne est définie par

$$d(n) = \begin{cases} 1 \text{ } \textit{si } n \textit{ a au moins une solution,} \\ 0 \text{ } \textit{autrement.} \end{cases}$$

Cette fonction en apparence très simple ne peut pas être *effectivement calculée*, c'est-à-dire calculée à l'aide d'une procédure effective [Matiiassevitch, 1995]. Plus précisément, cela signifie qu'il n'existe pas de méthode ou de règle générale permettant de calculer toutes les valeurs de d, et cela même s'il est possible d'en calculer certaines - on sait par exemple que l'équation $x - 1 = 0$ admet au moins une solution.

L'existence de fonctions non calculables par procédure effective est bien plus qu'un simple résultat mathématique : c'est un résultat sur les limites

en principe qui prouve qu'il existe des problèmes mathématiques - tel que celui de déterminer si une équation diophantienne possède une solution - dont la résolution échappe au calcul. La thèse de Church-Turing peut donc être à la fois considérée comme une *barrière* délimitant ce qui est calculable en principe et comme un argument très fort en faveur de la thèse selon laquelle les logiciens ont atteint un concept de calcul qui ne dépend pas de notre manière de définir ce que l'on entend par calculable :

> Il me semble que l'on a pour la première fois réussi à proposer une définition absolue d'une intéressante notion épistémologique, c'est-à-dire, d'une notion qui ne dépend pas du formalisme choisi [Gödel, 1946, p. 84].

La calculabilité proposée par Church et Turing est en effet une notion *absolue* pour les raisons suivantes :

1. **Elle ne dépend pas de la faisabilité pratique**. La calculabilité est centrée sur la notion de procédure ou règle effective dont l'exécution ne peut pas être limitée pour des raisons de faisabilité pratique. En d'autres mots,

   > [...] Tandis que l'itération d'une règle effective est toujours une procédure effective, l'exécution itérée d'une règle qui est physiquement faisable n'est généralement pas physiquement faisable [Dubucs, 2002, p. 222].

2. **Elle ne dépend pas du formalisme choisi**. Toutes les formalisations de la notion de procédure effective qui ont été proposées depuis 1930 se sont révélées être équivalentes au sens où elles définissent exactement la même classe de fonctions [Odifreddi, 1989].

3. **Elle ne dépend pas de la création de nouveaux ordinateurs**. Les limites du calcul d'un ordinateur reposent sur deux éléments principaux : sur la technologie de son matériel ou *hardware* et sur l'efficacité de son programme ou *software*. Les limites du calcul définies par la calculabilité sont en revanche indépendantes de ces deux éléments. D'une part, le matériel d'un ordinateur n'influe pas sur les fonctions calculables par procédure effective puisque la calculabilité est définie

> en faisant abstraction des ressources physiques. D'autre part, les programmes qui sont implémentés au sein des ordinateurs ne remettent pas en cause la calculabilité car les programmes informatiques *sont* des procédures effectives [Shepherdson and Strurgis, 1963].

La calculabilité est ainsi une véritable notion épistémologique au sens où elle permet d'acquérir une connaissance, celle de pouvoir déterminer ce qui est calculable, qui est absolue et intemporelle [Zwirn, 2000]. Autrement dit, la conception de nouveaux outils mathématiques ou de nouvelles technologies n'apporterait rien de nouveau en ce qui concerne les limites en principe préalablement établies par Church et Turing.

3 L'hyper-calcul : calculer au-delà de la barrière de Turing

Malgré la nature absolue des limites en principe, Copeland et Proudfoot ont émis en 1999 l'hypothèse selon laquelle la barrière de Turing pouvait être franchie [Copeland and Proudfoot, 1999]. Plus précisément, les auteurs ont proposé certaines idées provenant des travaux de Turing qui sont en faveur de la possibilité de calculer des fonctions non calculables par machine de Turing, possibilité aujourd'hui appelée *hyper-calcul*[8].

L'hyper-calcul soulève par définition deux questions majeures. Premièrement, est-ce que l'hyper-calcul est *logiquement* possible ? Autrement dit, peut-on formaliser des outils logiques, tels que la machine de Turing pour le calcul effectif, qui hyper-calculent ? Deuxièmement, est-ce que l'hyper-calcul est *physiquement* possible ? C'est-à-dire, peut-on constuire physiquement des dipositifs exécutant des hyper-calculs ? Une réponse positive à ces deux questions aurait ainsi pour conséquence de modifier en pronfondeur les limites du calcul considérées comme absolues depuis la première moitié du XX^{e} siècle.

8. L'hyper-calcul ne doit pas être confondu avec ce qui est communément appelé *calcul non conventionnel*. Tandis que l'hyper-calcul concerne les limites du calcul, le calcul non conventionnel concerne quant à lui les différents *paradigmes* du calcul [Cooper, 2013]. Ces paradigmes peuvent en particulier reposer sur l'utilisation de l'ADN [Păun et al., 1998], des membranes cellulaires [Păun, 2002] ou de la génétique [Michalewicz, 2002].

3.1 Origine conceptuelle de l'hyper-calcul

Pour Gödel, rappelons-le, la calculabilité est une notion absolue :

> Il me semble que l'on a pour la première fois réussi à proposer une définition absolue d'une intéressante notion épistémologique, c'est-à-dire, d'une notion qui ne dépend pas du formalisme choisi [Gödel, 1946, p. 84].

Toutefois et même si elle semble correcte, l'affirmation de Gödel n'est pas tout à fait complète :

> L'affirmation de Gödel [...] est seulement partiellement correcte : la définition de la notion de procédure effective ne dépend pas des détails du formalisme, mais repose de façon cruciale sur le fait que nous avons affaire en premier lieu à un formalisme. En ce sens, le caractère absolu de cette notion n'a été atteint seulement d'après une notion de formalisme élémentaire qui n'est pas expliquée [Sieg, 1997, p. 168].

Une possible interprétation de la remarque de Sieg consiste à dire que la notion de calcul effectif repose sur l'idée intuitve *d'opération élémentaire*. Pour Sieg, l'équivalence des différentes formalisations proposées par les logiciens des années 1930 a été un succès car l'on a réussi à se mettre d'accord sur les opérations élémentaires pouvant être exécutées lors d'un calcul. Par exemple, Turing justifie les opérations élémentaires de la machine de Turing - déplacement de la tête de lecture/écriture d'une cellule, effacement ou l'insertions d'un symbole, *etc.* - en expliquant qu'elles formalisent correctement la manière dont un mathématicien calcule :

> Il [Turing] considère ensuite les actions pouvant être exécutées par un être humain qui est en train de calculer ; il l'imagine en train de travailler sur du papier quadrillé à la manière d'un enfant travaillant sur un « livre d'arithmétique » [...] Le calcul est exécuté étape par étape sur un ruban constitué d'un nombre fini (mais arbitrairement grand) de cellules, pouvant contenir ou non un symbole. A chaque étape, l'action est déterminée par un

> ensemble fini d'instructions [...] Turing montre aisément que le comportement d'un calculator [un calculateur humain] peut être exactement simulé par une machine de Turing [Gandy, 1988, pp. 74-75].

C'est donc parce que les opérations élémentaires de la machine de Turing - et des autres formalismes - sont acceptées comme des opérations élémentaires que les limites du calcul sont absolues au sens où elles ne dépendent pas du formalisme choisi.

Ce dernier point suggère ainsi la thèse selon laquelle l'acceptation d'opérations qui ne sont généralement pas considérées comme élémentaires pourrait altérer l'équivalence entre ces formalismes et ainsi modifier les limites en principe. On trouve en particulier au sein du travail de Turing une formalisation pouvant mettre en doute l'affirmation de Gödel si ses opérations sont considérées comme élémentaires. Cette formalisation nommée *o-machine* a été introduite par Turing en 1939 dans sa thèse de doctorat :

> Supposons que l'on fournisse des moyens non spécifiés capables de résoudre des problèmes de théorie des nombres ; une sorte d'oracle. Nous n'irons pas plus loin concernant la nature de cet oracle si ce n'est que ce ne peut pas être une machine. A l'aide de l'oracle nous pouvons concevoir un nouveau type de machines (appelons-les *o*-machines), dont l'un des processus fondamentaux est de résoudre un problème de théorie des nombres donné [Turing, 1939, p.167].

Une *o*-machine (OM) est une machine de Turing qui possède un processus supplémentaire, l'appel de l'oracle. Cet oracle peut être considéré comme une boîte noire dont l'architecture n'est pas spécifiée et qui a la capacité de fournir les valeurs de certaines fonctions non calculables par machine de Turing. Plus précisément, si une fonction est calculable par machine de Turing l'oracle n'intervient pas ; mais dans le cas où la fonction n'est pas calculable, l'OM rentre dans un état interne particulier faisant intervenir l'oracle qui fournit la valeur de la fonction pour l'argument désiré.

La remarque de Sieg est ainsi fondamentale : si l'appel de l'oracle est considéré comme une opération élémentaire alors l'affirmation de Gödel selon laquelle la définition de la calculabilité est indépendante du formalisme choisi est fausse. Il existerait en particulier un formalisme - l'OM [9] - qui serait capable de calculer de manière effective des fonctions non calculables par machine de Turing.

Mais le calcul de l'OM n'est pas effectif car il ne peut pas être décrit à partir d'opérations élémentaires intuitives au même titre que les autres formalismes proposés dans les années 1930. Les opérations effectuées par une OM ne peuvent pas être qualifiées d'opérations élémentaires puisque l'on ne dispose que de très peu d'informations concernant l'appel de l'oracle ; on sait uniquement que l'OM rentre dans un état interne particulier et invoque l'oracle dans le but de fournir une valeur. Son fonctionnement ne peut donc pas être décrit à partir d'opérations élémentaires car ses opérations ne correspondent pas à l'idée intuitive que l'on a d'une opération élémentaire.

Ce point de vue est cohérent avec les travaux de Turing pour deux raisons :

1. Même si Turing ne fournit aucune précision quant à l'architecture d'une OM, ce dernier affirme néanmoins que son calcul ne peut pas être effectif :

   > Nous n'irons pas plus loin en ce qui concerne la nature de cet oracle mise à part que ce ne peut pas être une machine [de Turing] [Turing, 1939, p.167].

2. En introduisant l'OM, Turing n'avait pas pour but de montrer qu'il est possible de calculer des fonctions non calculables par machine de Turing. Plus exactement, il tentait d'évaluer la complexité des problèmes mathématiques. Son but était d'exhiber un problème mathématique qui n'était pas résoluble par les OM - capables par définition de résoudre des problèmes qui ne peuvent pas être résolus par procédure effective [Turing, 1939, p. 173].

Toutefois, même si l'OM n'a pas été créée dans le but d'hyper-calculer, Turing a bel et bien proposé un formalisme, qui n'est certes pas élémentaire,

9. Consulter [Cooper, 2004] pour une définition formelle de l'OM.

mais qui semble être capable de calculer davantage de fonctions que les formalismes traditionnels de type machine de Turing. De part son architecture et sa puissance calculatoire, il est par conséquent cohérent de considérer l'OM comme une piste en faveur de la possibilité de l'hyper-calcul.

3.2 L'hyper-calcul : mythe ou réalité ?

L'OM montre-t-elle qu'il est possible de franchir la barrière de Turing ? Si oui, remet-elle en cause les limites du calcul admises depuis les années 1930 ? Je pense qu'il serait prématuré de répondre affirmativement à ces deux questions car si l'OM peut être vue comme une première tentative en vue d'hyper-calculer, il est hautement improbable qu'elle permette de calculer des fonctions non calculables par machine de Turing.

En effet, le calcul d'une OM n'est pas à proprement parler un *hyper-calcul*. Dans sa définition la plus large, un hyper-calcul est un calcul qui produit les valeurs d'une fonction f non calculable par machine de Turing. Le problème est que si l'on se base sur cette définition, l'OM n'hyper-calcule pas. Tout ce que l'on sait au sujet de son fonctionnement est qu'il nécessite l'intervention de l'oracle ; comment l'oracle s'y prend-t-il pour hyper-calculer reste néanmoins un mystère. Certains chercheurs ont ainsi défendu que l'affirmation selon laquelle l'OM hyper-calcule est en fait une pétition de principe puisque l'oracle est par définition capable d'hyper-calculer [Davis, 2006]. En résumé, l'oracle n'est pas un moyen permettant de produire les valeurs d'une fonction non calculable par machine de Turing.

Le résultat précédent ne signifie pas pour autant qu'il soit impossible de franchir la barrière de Turing ; il prouve tout au plus que l'OM n'est pas le bon moyen pour y parvenir. Dit autrement, la conception de procédures non effectives ainsi que la construction physique de machines capables d'hyper-calculer ne sont pas remises en cause et restent par conséquent toujours possibles. Pour reprendre les mots de Davis,

> Comment pouvons-nous exclure à jamais la possibilité qu'un jour, un être (peut-être un visiteur extraterrestre) nous présente un dispositif (peut-être extrêmement complexe) ou "oracle" qui "calcu-

le" une fonction non calculable ? [Davis, 1958, p. 11]

Même si la question formulée par Davis semble être réthorique, il est possible d'y répondre. Apporter une réponse à cette question revient en effet à determiner si la construction physique d'un dispositif capable d'hyper-calculer - une *hyper-machine* - est *physiquement possible*, c'est-à-dire si elle est compatible avec les lois qui régissent l'univers. En particulier, si une telle construction se révèle être incompatible avec les lois physiques, la possibilité que l'on soit un jour confronté à une hyper-machine sera à exclure définitivement.

Néanmoins, autant l'impossibilité physique des hyper-machines remettrait en cause toute tentative dans le but de franchir la barrière de Turing, autant la possibilité physique des hyper-machines n'impliquerait pas que l'on puisse hyper-calculer. Hyper-calculer nécessite en effet de pouvoir produire les valeurs de fonctions non calculables, et la possibilité physique des hyper-machines ne permet pas de produire ces valeurs. Pour accéder à ces valeurs, il est en revanche nécessaire de *construire physiquement* une hyper-machine [10].

La possibilité de franchir la barrière de Turing exige par conséquent de démontrer la thèse suivante que j'appelle *thèse physique de l'hyper-calcul* :

Thèse (Thèse physique de l'hyper-calcul)
Il est possible de construire physiquement une hyper-machine.

La vérité de la thèse physique de l'hyper-calcul ne fait cependant pas l'unanimité parmi les scientifiques, ces derniers défendant trois positions différentes à propos de la construction physique d'une hyper-machine :

1. **La construction d'une hyper-machine est impossible.** Les adversaires de la thèse physique de l'hyper-calcul se divisent en deux catégories. Ceux de la première catégorie tentent de montrer que *certaines* propositions de construction sont vouées à l'échec [Douglas, 2003], [Hagar and Korolev, 2006]. Ceux de la seconde catégorie défendent en revanche que la construction d'une hyper-machine est impossible

10. La justification de cette affirmation fait l'objet du chapitre 2. En particulier, le lecteur trouvera dans l'introduction du chapitre 3 une définition précise de ce que j'entends par *construire physiquement une hyper-machine.*

quelle que soit la proposition examinée. Par exemple, Davis défend la thèse selon laquelle les lois physiques sont incompatibles avec une telle construction :

> C'est comme si le mot 'impossible' était vu comme un *challenge*. Bien que les lois de la thermodynamique ont démontré que la recherche d'une machine en mouvement perpétuel est un exercice futile, des inventeurs affirmant avoir construit de tels dispositifs assiègent encore les offices des brevets et réussissent à obtenir des soutiens financiers provenant d'investisseurs crédules [Davis, 2004, p. 195].

2. **La construction d'une hyper-machine est possible et permet d'hyper-calculer.** Il existe dans la littérature plusieurs propositions en vue de démontrer la thèse physique de l'hyper-calcul [11]. Même si les auteurs de ces propositions n'affirment pas que la technologie nécessaire pour construire une hyper-machine est actuellement disponible, ils défendent néanmoins que la construction de leur machine sera un jour possible.
3. **Même si la construction d'une hyper-machine est possible, elle n'est pas suffisante pour hyper-calculer**. Cette position signifie que la production de valeurs non calculables par une machine de Turing n'est pas suffisante pour hyper-calculer. Piccinini défend en particulier que le calcul de fonctions non calculables nécessite aussi de pouvoir identifier la fonction qui est calculée par l'hyper-machine, de pouvoir sélectionner la valeur que l'on souhaite calculer, *etc.* [Piccinini, 2011]. Si de tels critères peuvent sembler anodins, Piccinini défend qu'aucune des hyper-machines actuellement proposées ne les satisfait tous.

Malgré les divergences au sujet de la thèse physique de l'hyper-calcul, les critiques formulées à son encontre ne proviennent pas toutes du domaine physique mais ont aussi pour origine le domaine logique. Plus précisément, c'est la possibilité logique de l'hyper-calcul - l'énoncé selon lequel il est possible

11. Pour une vue d'ensemble de ces propositions, consulter [Copeland, 2002b], [Stannett, 2004] et [Stannett, 2006]. Ces propositions sont aussi détaillées dans le chapitre 3 de la présente thèse.

de calculer une fonction non calculable - qui est parfois visée. La possibilité logique de l'hyper-calcul est en effet fondamentale pour la thèse physique de l'hyper-calcul car un énoncé logiquement contradictoire ne peut pas être physiquement possible - *cf.* introduction du chapitre 1. Dans le détail, les arguments énoncés contre la possibilité logique de l'hyper-calcul ont généralement pour but de montrer que le calcul de fonctions non calculables implique des paradoxes. Ces paradoxes peuvent avoir deux origines différentes : les processus sur lesquels sont fondés les hyper-machines [Thomson, 1954] ou la possibilité de calculer certaines fonctions particulières [Cotogno, 2009]. Pour les défenseurs de l'hyper-calcul, le calcul de fonctions non calculables n'impliquent en revanche aucun problème logique [Copeland, 2002b], [Ord, 2002].

En résumé, déterminer si l'hyper-calcul permet de franchir la barrière de Turing exige de répondre à deux questions :

1. L'hyper-calcul est-il logiquement possible ?
2. La thèse physique de l'hyper-calcul est-elle vraie ?

Le présent travail est une tentative dans le but de répondre à ces questions.

4 Plan de l'argumentation

Mon travail a pour but de défendre une thèse philosophique sur les ordinateurs. Ma position est qu'il est indispensable d'étudier les ordinateurs pour établir les limites du calculable, que ces limites soient définies en principe ou en pratique. En ce sens, je remets en cause la distinction en principe / en pratique sur laquelle se fondent les logiciens des années 1930 pour établir des limites indépendantes de la construction de nouveaux ordinateurs.

Afin de soutenir cette position, j'évalue en particulier la possibilité de construire de nouvelles machines - appélés *hyper-machines* - capables d'hyper-calculer. La construction d'une hyper-machine aurait selon moi pour conséquence de modifier en profondeur les limites du calcul proposées dans les années 1930. En effet, ces limites - qui sont définies en principe, à savoir indépendamment des facteurs matériels - ne seraient plus pertinentes puisque

les hyper-machines pourraient calculer au-delà de ces limites. L'étude des ordinateurs deviendrait par conséquent indispensable pour définir les limites du calcul.

Ce travail est divisé en quatre chapitres. Le premier chapitre a une double vocation : il sert à la fois d'introduction à l'hyper-calcul et d'assise sur laquelle peut se construire mon argumentation. Plus précisément, je tente de montrer qu'il est logiquement possible d'hyper-calculer, autrement dit que l'hyper-calcul n'implique pas de contradiction. Une telle démonstration est fondamentale car la possibilité logique de l'hyper-calcul est une condition nécessaire à sa possibilité physique, et donc à la construction des hyper-machines.

Le deuxième chapitre constitue le cœur de mon argumentation car j'explique pourquoi la construction physique d'une hyper-machine pourrait remettre en cause la distinction en principe / en pratique sur laquelle sont fondées les limites établies par les logiciens des années 1930. D'après moi, une analyse en principe des limites du calcul - qui fait abstraction des ressources physiques - ne suffit plus pour définir ce qui est calculable. Précisément, je montre que cette analyse est adéquate pour définir les limites de certains calculateurs tels que ordinateurs actuels ou les êtres humains mais qu'elle est néanmoins insuffisante pour établir des limites objectives inhérentes à tout type de calculateur.

Dans le troisième chapitre, j'évalue les différentes tentatives proposées afin de démontrer la thèse physique de l'hyper-calcul - la thèse selon laquelle il est possible de construire physiquement une hyper-machine. Ces tentatives sont fondées sur diverses théories physiques telles que la physique newtonienne, la relativité générale ou la physique quantique. Je défends en particulier que l'une de ces propositions, à savoir celle utilisant l'aléatoire quantique, est plus prometteuse que les autres.

Le quatrième et dernier chapitre a quant à lui pour vocation de montrer les limites des thèses défendues dans les trois chapitres précédents. Je soulève plus précisément un problème épistémologique qui pourrait faire obstacle à la démonstration de la thèse physique de l'hyper-calcul. Ce problème - que je nomme *problème de la vérification* - est énoncé de la manière sui-

vante : supposons que l'on dispose d'un dispositif qui pourrait être une hyper-machine, peut-on vérifier qu'il hyper-calcule ? Mon but sera de montrer que le problème de la vérification n'admet à ce jour aucune solution convaincante. Si ce résultat négatif est correct, la démonstration de la thèse physique de l'hyper-calcul serait ainsi confrontée à une difficulté supplémentaire, celle de vérifier que l'on a réussi à construire une hyper-machine.

Abréviations

Voici les abréviations - classées par ordre alphabétique - qui sont utilisées tout au long de cette thèse :

- Hyper-machine : HM
- Machine de Turing : MT
- Machine de Turing accélérante : MTA
- Machine de Turing à oracle : OM
- Machine à oracle aléatoire : OMA
- Suite de nombres aléatoires : SNA
- *Supertask* : ST
- Thèse de Church-Turing : TCT
- Thèse de Martin-Löf-Chaitin : thèse MLC
- Thèse physique de l'hyper-calcul : TPHC

Chapitre 1

La possibilité logique de l'hyper-calcul

> In truth, Church and Turing claimed only that a universal Turing machine can match the behavior of any human mathematician working with paper and pencil in accordance with an algorithmic method- a considerably weaker claim that certainly does not rule out the possibility of hypermachines.
>
> Jack Copeland

Le but de ce chapitre est de défendre la possibilité logique de l'hyper-calcul. Par *possibilité logique de l'hyper-calcul*, j'entends la non contradiction logique de l'énoncé suivant : il est possible de calculer une fonction non calculable par machine de Turing (non Turing-calculable).

Pourquoi commencer ce travail par établir la possibilité logique de l'hyper-calcul alors que j'ai expliqué en introduction que les enjeux actuels liés à cette notion concernaient principalement sa possibilité physique ? Ma réponse est qu'il est nécessaire d'établir au préalable la possibilité logique de l'hyper-

calcul avant de pouvoir prouver sa possibilité physique.

Cette réponse peut paraître étonnante au premier abord car on considère généralement que la possibilité logique d'un énoncé n'implique pas sa possibilité physique, à savoir sa compatibilité avec les lois de la nature : un énoncé peut être logiquement possible et ne pas être physiquement possible. Par exemple, l'énoncé *un corps en mouvement possède une vitesse supérieure à celle de la lumière* n'est pas physiquement possible d'après la théorie de la relativité restreinte ; mais cet énoncé est pourtant logiquement possible puisque l'on peut concevoir mentalement, et sans contradiction logique, un corps se déplaçant à une vitesse supérieure.

En revanche, si un énoncé n'est pas logiquement possible alors il n'est pas physiquement possible. Ce point peut être illustré à partir de la trisection de l'angle, problème classique de mathématiques consistant à diviser un angle en trois parties égales à l'aide d'une règle et d'un compas. Enoncé sous cette forme, Pierre-Laurent Wantzel démontra à partir d'un raisonnement mathématique que le problème n'avait pas de solution [Wantzel, 1837]. Ce raisonnement permis en particulier d'exhiber une large famille d'équations de problèmes impossibles à résoudre à la règle et au compas. L'équation de la trisection de l'angle faisant partie de cette famille, il est donc physiquement impossible à l'aide d'une règle est d'un compas de diviser un angle en trois parties égales.

Par analogie avec le problème de la trisection de l'angle, si l'hyper-calcul se révélait logiquement contradictoire cela serait une preuve de son impossiblité physique. C'est pourquoi j'ai souhaité commencer ce travail en défendant la thèse selon laquelle l'hyper-calcul est logiquement possible.

Plus précisément, j'ai choisi de répondre tout au long de ce chapitre à deux des principales objections qui sont couramment énoncées contre la possibilité logique de l'hyper-calcul :

1. La première objection est énoncée contre la notion même d'hyper-calcul. Cette objection affirme que l'hyper-calcul est logiquement contradictoire car elle est en opposition avec la thèse de Church-Turing (TCT). L'objection peut être résumée comme ceci. Tout d'abord, la conjonction des énoncés *il est possible de calculer une fonction non Turing-*

calculable - énoncé de l'hyper-calcul - et *les fonctions calculables sont les fonctions Turing-calculables* - énoncé de la TCT - est contradictoire. Cette contradiction implique en particulier la vérité d'un seul des deux énoncés [1]. Enfin, puisque la TCT est admise par l'ensemble de la communauté scientifique, l'opposition conduit davantage à exclure l'hyper-calcul. L'objection montre ainsi que hyper-calcul est logiquement contradictoire en vertu de la vérité de la TCT.

2. La seconde objection est quant à elle invoquée contre les processus utilisés par les hyper-machines (HM) pour calculer des fonctions non Turing-calculables. De tels processus, que je détaillerai au moment de répondre à la première objection, reposent principalement sur l'exécution d'une infinité d'étapes de calcul en un temps fini. Les adversaires de l'hyper-calcul défendent ainsi que l'utilisation de processus infinis exécutés en un temps fini est logiquement contradictoire. Ces derniers exploitent pour cela deux arguments : un argument fondé sur les paradoxes logiques de l'infini et un argument s'appuyant sur les preuves de Turing montrant qu'une machine théorique pouvant calculer sa propre fonction arrêt ou sa propre fonction diagonale est contradictoire.

Commençons tout de suite par la première objection.

1 Calcul effectif *vs* hyper-calcul

Dans cette section, je réponds à une première objection qui a été énoncée contre la possibilité logique de l'hyper-calcul. Cette objection consiste à dire (1) que l'hyper-calcul est une notion en opposition avec la TCT et (2) que l'hyper-calcul est logiquement contradictoire en vertu de la vérité de la TCT.

La position selon laquelle la TCT - thèse affirmant que les fonctions intuitivement calculables sont Turing-calculables - est en opposition avec l'hyper-calcul - terme dénotant le calcul de fonctions non Turing-calculables - a tout d'abord pour origine un point de vue répandu dans la littérature qui consiste

1. En effet, si A et B sont deux énoncés dont la conjonction est fausse alors A ou - exclusif - B est vrai. Ce raisonnement est formalisé par le théorème de la logique propositionnel suivant : $(\neg(A \wedge B) \rightarrow \neg A \vee \neg B)$.

à étendre la portée de la TCT à d'autres procédures de calcul [Copeland, 2002a, p. 7]. Certains auteurs, qu'ils soient pour ou contre l'hyper-calcul, considèrent en effet la TCT comme une thèse énonçant que les fonctions calculables par *tous types de procédures* sont calculables par MT. Citons en particulier les passages frappant de Newell et des Churchland :

> Le fait qu'il existe une formulation plus générale de la notion de machine et que cette formulation conduise à un unique ensemble de fonctions entrée-sortie est appelée thèse de Church [Newell, 1980, p. 150].

> La thèse de Church dit que tout ce qui est calculable est Turing-calculable. Si l'on suppose avec une certaine sécurité que ce que fait le cerveau est calculable, alors il peut en principe être simulé par un ordinateur [Churchland and Churchland, 1983, p. 6]

Ou bien ceux de Smolensky et de King,

> [...] la thèse de Church comme l'affirmation que la classe des calculs bien définis est englobée par la machine de Turing [Smolensky, 1988, p. 3].

> Peut-il exister, d'autre part, des processus computationnels qui ne peuvent pas être simulés par une machine de Turing ? La croyance qu'il n'existe pas de tels processus est la thèse de Church-Turing, aussi connu comme la thèse de Church [King, 1996, p. 381].

Même certains défenseurs de l'hyper-calcul tel que Syropoulos considèrent la TCT comme une thèse pouvant être généralisée à tout ce qui est calculable :

> L'hyper-calcul est [...] l'idée selon laquelle la thèse de Church-Turing, qui est supposée décrire ce qui est calculable et ce qui n'est pas calculable, pourrait ne pas être vraie [Syropoulos, 2008, p. 1].

En résumé, ces auteurs soutiennent que la TCT ne s'applique pas uniquement au calcul effectif mais aussi au calcul en général. De ce point de vue, le calcul de fonctions non Turing-calculables est en opposition avec la TCT.

L'objection consiste ensuite à montrer que l'hyper-calcul est logiquement contradictoire en vertu de la vérité de la TCT. Plus précisément, les partisans d'une opposition entre hyper-calcul et TCT - qu'ils soient en faveur ou contre l'hyper-calcul - soutiennent l'une de ces deux affirmations :

1. L'hyper-calcul n'est pas logiquement possible car la TCT est vraie.
2. L'hyper-calcul est logiquement possible car la TCT est fausse.

Pour les adversaires de l'hyper-calcul, il existe toutefois une dissymétrie entre ces deux affirmations car la TCT est unanimement admise contrairement à l'hyper-calcul. Autrement dit, puisque les deux affirmations reposent en dernier lieu sur la vérité la TCT et que cette dernière est admise par l'ensemble de la communauté scientifique, c'est la première affirmation qui tend à être considérée comme vraie.

Dans le but de répondre à cette objection et de défendre la possibilité logique de l'hyper-calcul, je vais proposer une troisième position où l'hyper-calcul n'est pas en opposition avec la TCT. Cette position - qui prend racine dans les travaux de Copeland - a en particulier deux vertus : unifier les notions de calcul effectif et d'hyper-calcul au lieu de les opposer et clarifier les différences structurelles entre les formalisations de la calculabilité telles que la MT et les HM.

1.1 La justification de la thèse de Church-Turing

Avant de présenter la position qui permet selon moi de faire disparaître l'opposition entre hyper-calcul et TCT, je souhaite expliquer pourquoi la TCT est admise par la majorité des scientifiques. Une telle explication est importante pour deux raisons. Elle est d'une part importante pour comprendre l'objection énoncée contre l'hyper-calcul car d'après cette dernière, c'est parce que la TCT est vraie que l'hyper-calcul est logiquement contradictoire. Elle permet d'autre part de soutenir ma position selon laquelle l'hyper-calcul peut être logiquement possible tout en préservant la vérité de la TCT.

La position standard des scientifiques à l'égard de la TCT est de considérer cette thèse comme vraie tout en admettant qu'elle est formellement indémontrable. Il existe toutefois des différences au sein cette position et certains

auteurs considèrent la TCT comme

1. indémontrable mais vraie [Turing, 1936] ;
2. partiellement démontrée [Shoenfield, 1967] ;
3. démontrée [Gandy, 1988], [Dershowitz and Gurevich, 2008] ;
4. prouvable mais pas encore démontrée [Shoenfield, 1993], [Folina, 1998].

Dans ce qui suit, je vais analyser ces positions dans le but de justifier la position standard selon laquelle il est fortement probable que la TCT soit vraie même si elle n'est pas démontrable mathématiquement. Je terminerai par expliquer pourquoi certains arguments énoncés contre la TCT sont erronnés.

1.1.1 L'origine d'une justification : l'appel direct à l'intuition

Un des premiers à défendre ce que l'on nomme aujourd'hui la *thèse de Church-Turing* fut Turing lui-même. Turing était toutefois conscient de l'obstacle suivant :

> Tous les arguments qui peuvent être invoqués sont fondamentalement destinés à faire appel à l'intuition, et par conséquent condamnés à demeurer insatisfaisants d'un point de vue mathématique [Turing, 1936, p. 135].

Cet obstacle a pour origine une conception traditionnelle de la TCT qui se caractérise par l'idée selon laquelle la thèse identifie deux concepts, l'un informel - celui de procédure effective - et l'autre formel - celui de machine de Turing. Plus précisément, la difficulté que l'on rencontre en cherchant une démonstration de la TCT peut être exprimée par le dilemme suivant [Zénil, 2006] :

- Soit on admet que l'on ne peut pas prouver l'équivalence $\alpha \equiv \beta$, où α est informel et β est formel.
- Soit on pense qu'il est possible de préciser α en une nouvelle notion α' avec laquelle on pourra démontrer l'équivalence $\alpha' \equiv \beta$. Toutefois, ce que l'on démontre dans ce cas est l'équivalence $\alpha' \equiv \beta$ et non l'équivalence $\alpha \equiv \beta$. La thèse peut alors se reformuler comme $\alpha \equiv \alpha'$ et l'on revient au point précédent.

Bien qu'il semble impossible de démontrer la TCT en raison de la nature opposée des deux concepts contenus dans cette thèse, un argument fondé sur l'intuition a néanmoins été proposé par Turing dans le but de la justifier.

Cet argument consiste à identifier la notion de *fonction calculable par procédure effective* avec celle de *fonction calculable par un mathématicien idéalisé utilisant des moyens finis* :

> Les nombres [2] calculables sont les nombres dont les décimales sont calculables par des moyens finis [...] Nous pouvons comparer un homme en train de calculer [...] avec une machine [Turing, 1936, p. 117].

La stratégie de Turing n'a pas été de définir ce qu'est une fonction calculable, mais de déterminer quels sont les processus nécessaires pour exécuter un calcul effectif. Dans ce but, Turing énumère dans un premier temps les actions exécutées par un calculateur humain et montre dans un second temps que ces actions sont exécutables par une MT.

Turing commençe par définir le cadre de travail d'un mathématicien idéalisé sous certains aspects. Il explique qu'un calcul est habituellement exécuté en inscrivant certains symboles sur du papier que l'on peut supposer être à une dimension. De plus, il défend - très brièvement néanmoins - que le papier peut prendre la forme d'un ruban *arbitrairement grand* divisé en cellules. Tâchons de clarifier ce point. Le calcul est défini par Turing comme étant exécuté en principe, c'est-à-dire indépendamment de la faisabilité pratique. Précisément, du moment que la quantité de ressources est finie, le ruban arbitrairement grand permet au calculateur humain de disposer de toutes les ressources physiques dont il a besoin pour mener à bien son calcul. Par exemple, il peut toujours se réapprovisionner d'une nouvelle feuille s'il manque de papier sur quoi écrire.

Turing énumère ensuite les caractéristiques d'un calcul exécuté à l'aide de moyens finis :

2. A première vue, l'article de Turing traite des nombres réels calculables c'est-à-dire des nombres réels dont les décimales peuvent être produites par une MT. Cependant, Turing était en réalité intéressé par les fonctions de $\mathbb{N}^p \to \{0,1\}$ calculables [Gandy, 1988, p. 74].

1. Chaque cellule ne peut contenir qu'un nombre fini de symboles.
2. Le calculateur humain ne peut observer qu'un nombre fini de symboles.
3. Il ne peut disposer que d'un nombre fini d'états mentaux.
4. Le comportement du calculateur est déterminé à chaque instant par les symboles qu'il est en train d'observer et par son état mental à cet instant.
5. Le calcul ne peut comporter qu'un nombre fini d'instructions.
6. Le calcul peut être divisé en opérations élémentaires « comprenant toutes celles qui sont utilisées pour le calcul d'un nombre » [Turing, 1936, p. 118], à savoir
   (a) le possible changement d'un symbole sur l'une des cellules observées ;
   (b) le possible échange du contenu de deux cellules adjacentes ou séparées par un nombre fini de cellules ;
   (c) le possible changement d'état mental au cours des deux opérations précédentes.

Pour finir, Turing démontre que le comportement du calculateur humain qu'il a défini peut être exactement reproduit par une MT.

L'analyse de la notion de procédure effective que propose Turing apporte un argument convaincant en faveur de la TCT car elle suppose uniquement de la part de l'être humain la capacité de pouvoir reconnaître les symboles observés. Plus encore selon Gandy, l'analyse de Turing fait davantage que fournir un argument en faveur de la TCT, elle prouve un théorème :

> Toutes les fonctions qui peuvent être calculées par un être humain abstrait qui suit un ensemble fini d'instructions sont effectivement calculables par une machine de Turing et inversement [Gandy, 1988, p. 77].

Plus exactement, la preuve donnée par Turing est pour Gandy aussi rigoureuse que la plupart des preuves mathématiques habituellement acceptées. Les différences de positions vis-à-vis de la TCT ne viendraient pas de problèmes liés à la démonstration de la TCT mais auraient pour origine

le sujet traité par cette thèse - celui des limitations du calcul effectif. On peut néanmoins rester sceptique quant à l'argument de Gandy pour la raison précise qu'à l'heure actuelle la TCT fait encore débat au sujet de sa possible démonstration. Il est néanmoins possible de se mettre d'accord sur une démonstration partielle de la TCT.

1.1.2 La démonstration partielle de la TCT

La TCT est généralement énoncée en ces termes : les fonctions calculables par procédure effective sont Turing-calculables. La réciproque de cette affirmation - les fonctions Turing-calculables sont calculables par procédure effective - n'est en revanche presque jamais utilisée. La raison en est que cette réciproque est considérée comme la direction *facile* de la TCT et l'énoncé original comme la direction *difficile*.

La réciproque est considérée comme la direction facile car il est possible de la démontrer mathématiquement. Shoenfield a en effet démontré la direction facile à partir du raisonnement suivant [Shoenfield, 1967].

Shoenfield commence par définir les fonctions primitives récursives [3] comme étant la plus petite classe de fonctions qui

R1 contient les fonctions projection, addition, multiplication et la fonction caractéristique de $<$,

R2 est close sous composition,

R3 est close sous l'opération de minimisation - *cf.* annexe B.

Shoenfield explique ensuite que les fonctions initiales mentionnées dans **R1** sont clairement calculables et que les opérations de composition et de minimisation appliquées à des fonctions calculables produisent des fonctions calculables. Il procède enfin comme ceci :

> Afin de prouver que chaque fonction récursive à la propriété P, il suffit de prouver que **R1**, **R2**, **R3** restent vraies lorsque nous remplaçons une fonction récursive par une autre fonction ayant

3. Shoenfield utilise dans sa démonstration la formalisation des fonctions récursives qui est équivalente à celle de la MT. Je présente cette formalisation dans l'annexe B.

> la propriété P. Une telle preuve est appelée une preuve par induction sur les fonctions récursives. En utilisant ce type de preuve, nous pouvons prouver de la même manière que chaque fonction récursive est calculable [Shoenfield, 1967, p. 109].

La direction difficile n'a quant à elle pas été démontrée mais il existe un puissant argument en sa faveur. Si l'argument a été présenté pour la première fois par Church [Church, 1936, p. 100], je me repose une fois de plus sur la présentation de Shoenfield qui est d'après moi plus accessible.

Ce dernier considère le calcul d'une fonction unaire f composée d'un nombre fini de configurations - par exemple de quadruples de MT - chacune codée par un entier a_i, où a_0 encode l'argument de la fonction et a_n le résultat. En utilisant $\langle a_0, \ldots, a_i \rangle$ pour désigner le code de la suite de configurations $(a_0, \ldots, a_i)$, il explique l'argument de Church :

> Maintenant la procédure effective [4] nous dit comment nous pouvons déduire a_i à partir de $\langle a_0, \ldots, a_{i-1} \rangle$. Par conséquent, il existe une fonction calculable g telle que $g(\langle a_0, \ldots, a_{i-1} \rangle) = a_i$. La procédure effective nous permet aussi de savoir à quel moment le calcul se termine, par conséquent il existe un prédicat calculable P tel que $P(\langle a_0, \ldots, a_i \rangle)$ est faux pour $i < n$ et vrai pour $i = n$ [Shoenfield, 1967, pp. 120-121].

L'argument de Church n'est pas une preuve de la direction difficile de la TCT. Comme le remarque en effet Church, f est Turing-calculable si on fait l'hypothèse selon laquelle g et P sont eux aussi Turing-calculables. Ainsi

> Notre tentative pour définir la calculabilité est de nature circulaire, puisque g et P doivent être supposés Turing-calculables. Toutefois, puisque g décrit une seule étape de calcul, c'est une fonction Turing-calculable très simple ; et il en va de même pour P. Nous pouvons donc supposer que g et P sont Turing-calculables.

4. Shoenfield cherche une preuve de la partie difficile de la TCT, c'est-à-dire une preuve de l'énoncé suivant : si une fonction est calculable par une procédure effective alors elle est Turing-calculable. Puisque cet énoncé prend la forme d'une implication, Shoenfield commence par supposer l'antécédent pour en déduire le conséquent.

> Si nous le supposons, alors nous pouvons prouver que f est Turing-calculable [Shoenfield, 1967, pp. 120-121].

De plus, même si la partie difficile de la TCT n'est pas prouvable à partir de l'argument de Church, Shoenfield reste confiant quant à la possibilité d'en fournir une démonstration. Il écrit à ce propos que

> Puisque la notion de fonction calculable n'a pas été définie précisément, il peut sembler impossible de fournir une démonstration de la thèse de Church-Turing. Cependant, ce n'est pas nécessairement le cas. Nous comprenons la notion de fonction calculable de façon assez précise pour forger certaines affirmations à son propos. Autrement dit, nous pouvons énoncer certains axiomes concernant les fonctions calculables qui seraient admis comme étant vrais par la plupart des gens. Il pourrait être possible de prouver la thèse de Church-Turing avec de tels axiomes. Toutefois, personne n'a réussi à succéder à cette tâche (bien que certains résultats intéressants ont été obtenus) [Shoenfield, 1993, p. 26].

Aujourd'hui encore, aucune preuve de la partie difficile n'a vu le jour. Ce sont les difficultés rencontrées afin de proposer une telle preuve qui ont poussé les chercheurs à considérer la partie difficile de la TCT comme étant la véritable thèse de Church-Turing, la partie facile étant aisément démontrable. Je vais maintenant expliquer deux arguments fondamentaux en faveur de la TCT, arguments servant à justifier la position selon laquelle la MT est une formalisation adéquate de la notion de procédure effective.

1.1.3 Argument de convergence et absence de contre-exemple

L'argument de convergence. Cet argument peut être énoncé comme ceci : puisque toutes les tentatives proposées dans le but de formaliser la notion de procédure effective se sont révélées équivalentes, il existe une très forte probabilité que ces tentatives caractérisent de façon adéquate cette notion.

Plus précisément, les formalisations - de la notion de procédure effective - peuvent être équivalentes de deux points de vue différents :

1. D'un point de vue *extensionnel*, deux formalisations sont équivalentes lorsqu'elles définissent exactement les mêmes fonctions.
2. D'un point de vue *intensionnel*, deux formalisations sont équivalentes si chaque procédure effective exprimée dans un formalisme peut être exprimée dans l'autre.

L'argument de convergence repose sur le fait que les formalisations dont on dispose sont équivalentes en extension :

> Les caractérisations de Turing et de Kleene, aussi bien que celles de Church, Post, Markov et d'autres encore, sont toutes équivalentes ; c'est-à-dire, exactement les mêmes fonctions sont obtenues pour chacune d'elles [Rogers, 1987, p. 18].

Mais aussi en intension :

> Pour chacune des caractérisations, nous pouvons montrer qu'il existe une procédure effective pour laquelle, étant donné n'importe quel ensemble d'instructions P à partir de la première caractérisation, nous pouvons trouver un ensemble d'instructions P provenant de la seconde caractérisation, pour lesquelles la même fonction est calculée [Rogers, 1987, p. 19].

Cet argument amena ainsi Church à écrire le passage suivant :

> Le fait que deux formalisations si différentes représentant de manière naturelle la notion de procédure effective s'avèrent être équivalentes, renforce d'autre part les raisons de croire qu'elles constituent une caractérisation générale de cette notion qui est cohérente avec l'idée intuitive que nous en avons [Church, 1936, p. 90].

L'absence de contre-exemple. Il n'existe actuellement pas de contre-exemple de la TCT. Cela signifie que toutes les fonctions calculables que l'on connaît peuvent être calculées par une MT. Cet argument peut sembler à première vue anodin puisque l'absence de contre-exemple est un prérequis si l'on souhaite argumenter en faveur de la TCT. Néanmoins, l'absence d'un tel contre-exemple est loin d'être trivial.

Les logiciens ont en effet été confrontés dans le passé à un contre-exemple. La fonction qui fut exhibée ne venait pas falsifier la TCT proprement dite mais une autre thèse moins générale : les fonctions calculables par procédures effectives sont les fonctions primitives récursives [5].

Avant d'exhiber le contre-exemple en question, à savoir une fonction qui n'est pas primitive récursive mais néanmoins calculable par procédure effective, il est utile de montrer dans un premier temps qu'une telle fonction existe.

L'existence de fonctions non primitives récursives mais néanmoins calculables peut être établie *via* une *méthode de diagonalisation* [Wolper, 2006, p. 130]. L'ensemble PR des fonctions primitives récursives étant en bijection avec $\mathbb{N}$, on peut énumérer ses éléments comme suit : $f_0, f_1, f_2, \ldots$ Et puisque ces fonctions peuvent avoir un nombre arbitraire d'arguments, on notera $f_i(n)$ la valeur

$$f_i \underbrace{(n, \ldots, n)}_{p \text{ fois}}$$

si $f : \mathbb{N}^p \to \mathbb{N}$. Maintenant considérons le tableau A suivant pour lequel l'élément $A_{i,j}$ contient la valeur $f_i(j)$:

A	0	1	2	$\ldots$	j	$\ldots$
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$\ldots$	$f_0(j)$	$\ldots$
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$\ldots$	$f_1(j)$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
f_i	$f_i(0$	$f_i(1)$	$f_i(2)$	$\ldots$	$f_i(j)$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\ddots$

On définit enfin la fonction

$$g(n) = f_n(n) + 1 = A_{n,n} + 1$$

Cette dernière fonction est calculable mais n'est pas primitive récursive. Tout d'abord, si g était primitive récursive, elle devrait être égale à f_k pour

5. Les fonctions primitives récursives forment un sous-ensemble propre des fonctions récursives - *cf.* annexe B. Cela signifie que l'ensemble des fonctions primitives récursives ne contient pas toutes les fonctions intuitivement calculables.

un certain k ; on devrait dès lors avoir $g(k) = f_k(k)$. Or par définition, $g(k) = f_k(k) + 1$. La fonction g n'est donc pas primitive récursive.

Si cette fonction n'est pas primitive récursive, elle est en outre calculable par procédure effective pour les raisons suivantes :

1. On peut énumérer l'ensemble des fonctions primitives récursives comme suit : on enumère toutes les chaînes de caractères sur un alphabet fini adéquatement choisi et on ne garde que celles qui correspondent à la définition d'une fonction primitive récursive. On se convainc de l'existence d'un tel algorithme car il est aisé de déterminer si une chaîne de caractères représente une définition valide de fonction. Il suffit pour cela de satisfaire certaines règles de syntaxe facilement exprimables. Par exemple, aucun préfixe ne peut contenir plus de parenthèses ouvrantes que fermantes, *etc.*
2. Une fois la fonction f_n obtenue, on l'évalue en n, ce qui est toujours possible car f_n est primitive récursive donc calculable.
3. Enfin, on calcule $f_n(n) + 1$ pour obtenir la valeur de $g(n)$.

L'existence de fonctions calculables non primitives récursives a ensuite permis de définir explicitement un de ces fonctions, à savoir la *fonction d'Ackermann* [Ackermann, 1928] :

Définition (Fonction d'Ackermann)
La fonction d'Ackermann $\mathcal{A} : \mathbb{N}^2 \to \mathbb{N}$ *est définie par*

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

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

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

La fonction d'Ackermann est un véritable contre-exemple de la thèse selon laquelle les fonctions calculables sont les fonctions primitives récursives. Ce contre-exemple a obligé les logiciens à étendre la notion de fonction calculable par procédure effective aux fonctions récursives. Les fonctions récursives sont en particulier définies en introduisant un nouveau schéma, à savoir le schéma de minimisation non bornée :

Définition (Schéma de minimisation non bornée)
Soit $A \subseteq \mathbb{N}^{p+1}$. On définit la fonction $f : \mathbb{N}^p \to \mathbb{N}$ comme

$$f(x_1, \ldots, x_p) = \begin{cases} 0 \text{ si } \{y/(x_1, \ldots, x_p, y) \in A\} = \emptyset \\ \text{le plus petit } y/(x_1, \ldots, x_p, y) \in A \text{ autrement.} \end{cases}$$

On note cette fonction $f(x_1, \ldots, x_p) = \mu_y((x_1, \ldots, x_p, y) \in A)$.

Cette définition est toutefois problématique car si on n'y prend garde, on peut construire des fonctions non calculables. En particulier, il se peut que l'on obtienne jamais la valeur de $f(x_1, \ldots, x_p)$. Imaginons par exemple que l'on souhaite calculer

$$f(x_1, \ldots, x_p) = \mu_y((x_1, \ldots, x_p, y) \in A)$$

Pour rechercher le plus petit entier y tel que $(x_1, \ldots, x_p, y) \in A$, on commence par tester si $(x_1, \ldots, x_p, 0)$ appartient à A. S'il n'appartient pas à A, on considère alors $(x_1, \ldots, x_p, 1)$ et ainsi de suite. Si on trouve une valeur de y satisfaisant $(x_1, \ldots, x_p, y) \in A$, la procédure s'achève ; sinon, on n'obtiendra jamais une telle valeur. C'est dans ce dernier cas que la fonction n'est pas calculable car puisqu'on ne fixe aucune borne à ne pas dépasser contrairement au schéma de minimisation bornée.

Dans le but de contourner ce problème, les logiciens ont donc introduit une nouvelle définition :

Définition (Partie sûre)
Une partie $A \subseteq \mathbb{N}^{p+1}$ est dite sûre si

$$\forall (x_1, \ldots, x_p) \in \mathbb{N}^p, \exists y \in \mathbb{N}/(x_1, \ldots, x_p, y) \in A.$$

Les fonctions récursives sont enfin définies comme ceci :

Définition (Fonctions récursives)
L'ensemble des fonctions récursives est le plus petit sous-ensemble de fonctions définies sur $\mathbb{N}$ qui contient les fonctions primitives récursives et qui est clos pour la minimisation non bornée de parties sûres.

A ce jour, aucune fonction calculable par procédure effective qui n'est pas récursive n'a été exhibée. Mieux encore, la méthode de diagonalisation ne permet pas de prouver l'existence d'une telle fonction :

> Lorsque Church proposa sa thèse, je m'assis afin de la falsifier en la diagonalisant hors de la classe des fonctions λ-définissables. Mais, réalisant rapidement que la diagonalisation ne pouvait être faite de façon effective, je devins cette nuit là un partisan de la thèse [Kleene, 1981, p. 59].

Le lecteur est à présent capable, à partir des arguments en faveur de la TCT qui viennent d'être présentés, d'entrevoir pourquoi l'hyper-calcul pourrait être considéré comme étant faux si on l'oppose à la TCT. Pour ma part, je pense que cette opposition n'a pas lieu d'être même si je suis d'accord avec la thèse selon laquelle la TCT est vraie. Il serait cependant injuste d'expliquer en détails ma position sans examiner au moins un des arguments qui ont été énoncés contre la TCT.

Certains auteurs ont en effet cherché à montrer que la TCT est fausse. En particulier, Bowie a publié un article en 1973 intitulé *An argument against Church's thesis* dans lequel il défend que la TCT peut être falsifiée dans la direction facile[6]. Toutefois, l'argument de Bowie est selon moi fallacieux et ne remet pas du tout en cause la vérité de la TCT.

1.1.4 Le contre-exemple de Bowie

Pour Bowie, affirmer qu'une fonction f unaire est calculable c'est affirmer qu'il existe un algorithme A pouvant calculer $f(x)$ pour tout argument x appartenant au domaine de f[7]. Plus précisément, si $\Diamond$ signifie *il est en principe possible que*, une fonction est calculable si

(I) $\Diamond\exists A\forall x[A$ peut calculer $f(x)$ avec x pour donnée en entrée$]$

6. Bowie n'est pas le seul à argumenter contre la TCT. Même si Peter soutient comme Bowie que la direction facile est fausse [Peter, 1959], Kalmar défend quant à lui la thèse selon laquelle la classe des fonctions récursives ne contient pas toutes les fonctions calculables, autrement dit que la direction difficile de la TCT est fausse [Kalmar, 1959].

7. Bowie n'utilise que des fonctions unaires dans son argument mais il serait aisé de l'étendre à des fonctions n-aires.

Expliquons cette définition. D'une part A est un algorithme, c'est-à-dire une procédure effective qui permet de calculer $f(x)$ pour tout x appartenant au domaine de f. D'autre part, l'expression *en principe possible* signifie deux choses. La première est que les ressources nécessaires au calcul sont en quantités arbitrairement grandes mais finies. La seconde est que l'algorithme n'a pas besoin d'être exhibé afin de prouver qu'une fonction est calculable ; il suffit de démontrer la possibilité d'un tel algorithme.

Cependant, l'expression A *calcule* $f(x)$ *pour l'entrée* x n'est pas assez précise selon Bowie. Suite à une longue analyse, l'auteur arrive ainsi à une meilleure définition : A calcule $f(x)$ pour l'entrée x si

(II) $\exists y$ [A établit que $f(x) = y$ où y est sous la forme d'une notation (codage) appropriée]

Mais ici encore, l'expression A *établit que* $f(x) = y$ est imprécise. L'auteur explique alors que l'algorithme A établit que $f(x) = y$ si et seulement si (1) la donnée en sortie de A sur x est $f(x)$, et (2) un être humain établit que $f(x) = y$. Bowie ne donne pas plus de détails sur l'expression *établir* mais on pourrait néanmoins dire que A établit que $f(x) = y$ s'il est possible - au sens de Bowie - qu'un être humain suive chaque instruction de A et arrive au résultat $f(x) = y$ à partir de x.

Pour résumer, une fonction f est calculable si les conditions suivantes sont satisfaites

(I) $\Diamond\exists A\forall x$[$A$ peut calculer $f(x)$ avec x pour donnée en entrée]

(II) $\exists y$ [A établit que $f(x) = y$ où y est sous la forme d'une notation appropriée]

A partir de cette définition, Bowie va exhiber une fonction qui n'est pas calculable mais récursive afin de montrer que la TCT est fausse dans la direction facile.

Il existe une fonction non calculable et récursive. Bowie commence par définir l'énoncé r comme étant un énoncé soit vrai soit faux mais dont il est impossible de déterminer s'il est vrai ou faux. D'après l'auteur, l'énoncé

r pourrait par exemple être « César croyait que les baleines sont des mammifères » [Bowie, 1973, p. 69]. Bowie définit ensuite une fonction f avec $n \in \mathbb{N}$ telle que

$$f(n) = \begin{cases} 1 \text{ si } r \text{ est vrai} \\ 0 \text{ si } r \text{ est faux} \end{cases}$$

Selon Bowie, f n'est pas calculable mais est récursive. Pour montrer que f n'est pas calculable, l'auteur raisonne dans un premier temps par l'absurde. Si f est calculable alors d'après **(I)** et **(II)** il existe un algorithme qui permet à un être humain d'établir pour tout n, un y tel que $f(n) = y$. L'être humain pourrait ainsi établir pour tout n si $f(n) = 0$ ou si $f(n) = 1$, c'est-à-dire déterminer si r est vrai ou faux. Toutefois, cette conclusion ne peut pas être correcte car elle rentrerait en contradiction avec l'hypothèse de départ selon laquelle la valeur de vérité de r ne peut pas être établie.

Bowie explique ensuite dans un second temps pourquoi f est résursive. Soient les fonctions constantes C_1 et C_0 définies pour tout $n \in \mathbb{N}$ par $C_1(n) = 1$ et $C_0(n) = 0$. Tout d'abord, puisque r est soit vrai soit faux, il suit que $f(n) = 0$ pour tout n ou $f(n) = 1$ pour tout n. Ainsi, $f = C_1$ ou $f = C_0$. En outre, il est clair que C_1 et C_0 sont toutes les deux des fonctions récursives. Par conséquent, d'après le théorème

$$\forall h \forall g[(h = g \text{ et } h \text{ est récursive}) \rightarrow g \text{ est récursive}]$$

f est récursive - en prenant $h = C_1$ ou C_2 et $g = f$. En résumé, la fonction f est selon Bowie une authentique fonction récursive non calculable, résultat qui l'amène à affirmer que la direction facile de la TCT est fausse.

Objections. La principale objection contre le résultat de Bowie a été proposée par Ross [Ross, 1974]. Cette objection a pour but de critiquer la définition de fonction calculable proposée par Bowie qui suppose l'existence d'un algorithme permettant à un être humain d'établir $f(x)$ pour tout x.

Pour Ross [8],

(III) Une fonction f définie sur les entiers naturels est calculable si et seulement s'il existe un algorithme A tel que, étant donné un $Gn\ \ulcorner n \urcorner$, A appliqué à $\ulcorner n \urcorner$ conduit au $Gn\ \ulcorner m \urcorner$ si et seulement si $f(n) = m$.

Cette définition a une signification différente de celle de Bowie car elle ne fait pas référence au fait qu'un être humain doit établir que $f(n) = m$. Pour Ross, le calcul algorithmique est une simple manipulation de symboles à partir d'un ensemble de règles précises qui ne nécessite aucune capacité humaine - établir, connaître, posséder - autre que la simple capacité de manipuler les symboles en suivant les instructions. D'après cette définition, la fonction

$$f(n) = \begin{cases} 1 \text{ si } r \text{ est vrai} \\ 0 \text{ si } r \text{ est faux} \end{cases}$$

est calculable, car l'une des deux fonctions C_1 ou C_2 conduit aux résultats de f. En effet, si deux personnes calculent en même temps C_1 et C_0, il suit que la fonction f est calculée par une des deux personnes même si aucune des deux n'établit qui calcule f. En d'autres termes, f est calculable indépendamment du fait de savoir quel est l'algorithme qui permet d'établir $f(n) = m$; plutôt, f est calculable car il existe une fonction récursive ayant le même comportement entrée-sortie.

Comme le remarque Mendelson, cette définition de la calculabilité suppose d'adopter une position classique qui n'est pas constructive car nous n'avons pas besoin de savoir lequel des deux calculateurs humains calcule f ; nous nous reposons uniquement sur le principe du tiers-exclu [Mendelson, 1990, p. 226]. Cependant, cette position non constructive ne remet pas en cause la TCT car d'après Mendelson c'est du point de vue classique que la TCT fut originellement proposée et discutée. L'auteur ne dénie pas pour autant la possible pertinence d'une position constructive au sujet de la notion de procédure effective.

8. Dans la définition qui suit, $Gn\ \ulcorner n \urcorner$ est le nombre de Gödel de l'entier n, c'est-à-dire le code attribué à n *via* la méthode utilisée par Gödel dans [Gödel, 1931]. Pour une présentation détaillée de ce codage, consulter l'annexe A.

Bien que l'argument de Bowie ne soit qu'un exemple parmi les arguments énoncés afin de falsifier la TCT, aucun des ces arguments n'a réussi a montrer que la TCT est fausse. Tout porte ainsi à croire que les fonctions calculables sont les fonctions calculables par MT.

Cette conclusion n'est cepedant qu'une première étape dans le but de montrer que l'hyper-calcul est logiquement possible. Plus précisément, tant que je n'aurai pas montré que la TCT n'est pas en opposition avec l'hyper-calcul, cette dernière notion sera considérée comme étant fausse en vertu de la TCT. Mon but est donc à présent d'expliquer de quelle manière il est possible d'unifier calcul effectif et hyper-calcul au sein d'une notion générale de calcul de sorte que leurs possibilités logiques soient compatibles.

1.2 Une tentative d'unification

La section précédente - présentation des fondements de la justification de la TCT - permet à présent de mieux comprendre pourquoi une opposition entre TCT et hyper-calcul remet en cause la possibilité logique de l'hyper-calcul.

Si l'on considère d'une part la TCT comme la thèse selon laquelle les fonctions calculables sont Turing-calculables et que l'on caractérise d'autre part l'hyper-calcul par l'énoncé selon lequel il est possible de calculer des fonctions non Turing-calculable, on arrive à la disjonction exclusive suivante : soit les fonctions calculables sont exclusivement les fonctions Turing-calculables, soit d'autres fonctions qui ne sont pas Turing-calculables sont calculables. Le problème est que si l'on soutient la pertinence de cette disjonction, les arguments en faveur de la TCT que je viens d'exposer condamnent l'hyper-calcul car elles favorisent de fait le premier membre de la disjonction. Le calcul de fonctions non Turing-calculables n'est donc pas logiquement possible si l'on considère qu'il existe une opposition entre TCT et hyper-calcul.

Dans le but de défendre la possibilité logique de l'hyper-calcul, je vais néanmoins montrer que les notions de calcul effectif et d'hyper-calcul ne s'opposent pas et qu'il est logiquement possible d'hyper-calculer tout en préservant la vérité de la TCT.

1.2.1 Le calcul effectif défini en termes de contraintes

Dans son manuel sur la calculabilité, Rogers définit une procédure effective ou algorithme de la manière suivante :

> Etant donnée un ensemble constitué de données en entrée représentées par une chaîne de symboles, une procédure effective est une procédure déterministe qui, appliquée à chacune de ces données en entrée, conduit éventuellement à une donnée en sortie correspondante, elle aussi représentée par une chaîne de symboles [Rogers, 1987, p. 1].

Un des problèmes inhérents à la notion de procédure effective réside dans son caractère imprécis qui est un obstacle à la démonstration de la TCT [9]. Néanmoins, Rogers parvient à préciser la notion d'algorithme en énonçant plusieurs caractéristiques qui lui apparaissent comme étant essentielles [Rogers, 1987, p. 2] :

1. Un algorithme est un ensemble fini d'instructions.
2. Un calculateur humain doit pouvoir réagir aux instructions et exécuter les calculs.
3. Le calculateur humain doit disposer de matériels - crayons, papier - afin d'exécuter chaque étape du calcul.
4. Soient P un ensemble d'instructions et L un calculateur humain. Pour toute donnée en entrée, L réagit à P *via* un processus discret, c'est-à-dire que le calcul est exécuté étape par étape.
5. L réagit à P *via* un processus déterministe sans faire appel à des dispositifs ou méthodes aléatoires.

Voici cependant une définition du concept d'algorithme proposée par Horowitz *et al.* qui est différente de celle de Rogers [Horowitz et al., 2007, p. 1]. Un algorithme est un ensemble fini d'instructions qui doit satisfaire les conditions suivantes :

9. Pour rappel, la contrainte principale qui pèse sur une possible démonstration de la TCT est qu'elle fait le lien entre une notion formelle - celle de MT - et une notion informelle - celle de procédure effective [Turing, 1936].

1. L'algorithme doit comporter une quantité arbitraire mais finie de données en entrée.
2. L'algorithme doit conduire à au moins une donnée en sortie.
3. Chaque instruction doit être claire et non ambigue.
4. Le nombre d'étapes de calcul nécessaire afin d'exécuter l'algorithme doit être fini.
5. En plus de la condition 3, chaque instruction doit pouvoir être exécutée par une personne n'utilisant que des crayons et du papier.

Les conditions énoncées au sein de la définition d'Horowitz *et al.* sont quasiment identiques aux caractéristiques incluses dans celle de Rogers exception faite du caractère discret et déterministe de la procédure. La raison de l'omission du caractère discret de la procédure provient d'une des distinctions entre mathématiques et informatique. Contrairement aux mathématiciens qui manipulent le continu en analyse, les informaticiens étudient plus fréquemment les mathématiques discrètes puisque les ordinateurs actuels sont des machines digitales. Par conséquent d'un point de vue informatique la condition de discrétisation n'est pas nécessaire.

Il n'en va toutefois pas de même concernant le caractère déterministe de la procédure dont l'omission provient quant à elle de la divergence des points de vue des auteurs. Fidèle à l'analyse de Turing, Rogers considère que l'idée de déterminisme est ancrée dans la définition intuitive de procédure effective au sens où les processus aléatoires - lancer de dés équilibrés par exemple - ne sont généralement pas utilisés lors de l'exécution d'un calcul :

> Le comportement du calculateur est déterminé à chaque instant par les symboles qu'il est en train d'observer et par l'état mental dans lequel il se trouve à cet instant [Turing, 1936, p. 136].

D'un autre côté, Horowitz *et al.* ne raisonnent pas en termes de conditions intuitives mais en termes de fonctions calculables. Ces derniers n'ont pas inclus la condition de déterminisme dans leur définition car cette condition est contingeante à la notion d'algorithme du point de vue des fonctions calculables. Plus précisément, l'ajout ou la suppression du déterminisme ne modifie pas la classe des fonctions calculables par les formalisations de la

notion d'algorithme. La preuve en est qu'une MT déterministe - identique à une MT standard - calcule la même classe de fonctions qu'une MT non déterministe.

Il est en effet possible de simuler le fonctionnement d'une MT par une *machine de Turing probabiliste* (MTP). Les MT que j'ai considérées jusqu'à présent sont déterministes puisqu'à partir d'une configuration donnée, il existe au plus une configuration possible pour l'étape suivante. Une MTP est quant à elle identique à une MT déterministe excepté en ce qui concerne son programme. Dans le cas d'une MT déterministe, le programme de la machine peut être défini comme une fonction de la forme

$$Q \times S \rightarrow Q \times S \times \{R, L\}$$

qui siginifie que le programme définit pour chaque paire (état de la machine, symbole lu) au plus un seul triplet (état suivant, symbole à écrire, direction de la tête de lecture). De son côté, le programme de la MTP n'est pas une fonction mais une relation Δ de la forme

$$Q \times S \subseteq Q \times S \times \{R, L\}$$

qui ne définit pas un seul triplet (état suivant, symbole à écrire, direction de la tête de lecture) mais un ensemble de tels triplets. Intuitivement, la MTP peut choisir n'importe lequel de ces triplets pour calculer la valeur en sortie d'un argument donné. Une fonction $f(x_1, \ldots, x_n)$ est dite calculable par MTP si, pour chacun des arguments $(x_1, \ldots, x_n)$ du domaine de f, il existe *au moins* une suite de configurations exécutée par la machine conduisant au résultat $f(x_1, \ldots, x_n)$. Enfin, il a été démontré que toute fonction calculable par une MTP est calculable par une MT et inversement [Hopcroft et al., 1979, p. 164].

Les définitions de Rogers et d'Horowitz *et al.* illustrent ainsi deux manières de définir le concept de procédure effective : l'une fondée sur les caractéristiques intuitives de la notion d'algorithme et l'autre sur les conditions nécessaires pour garantir la classe des fonctions Turing-calculables. C'est cette seconde

manière de procéder qui permet de contourner le problème de l'opposition entre calcul effectif et hyper-calcul.

1.2.2 Conséquences pour la notion d'hyper-calcul

Fonder la notion d'algorithme - ou de procédure effective - sur les caractéristiques qui sont nécessaires pour conserver la classe des fonctions Turing-calculables est la solution qui permet de montrer que le calcul effectif et l'hyper-calcul ne sont pas opposés. Ces deux notions doivent au contraire être considérées comme des *définitions alternatives du calcul*.

La solution au problème de l'opposition entre calcul effectif et hyper-calcul réside plus précisément dans la définition que donne Copeland de la notion de procédure effective [Copeland, 2002b, p. 1]. Une M est d'après Copeland une méthode de calcul devant satisfaire les contraintes suivantes :

1. M doit avoir un nombre fini de symboles et d'instructions.
2. M doit contenir un nombre fini d'étapes.
3. M doit pouvoir être exécuté dans un temps fini.
4. Un être humain doit pouvoir suivre M étape par étape, de la donnée initiale au résultat indépendamment des contraintes de temps et d'espace mémoire.
5. M doit pouvoir être exécuté par un être humain sans l'aide d'aucune machine physique.
6. Un être humain doit pouvoir exécuter chaque étape de M de manière *mécanique*, c'est-à-dire sans ingéniosité ni intelligence.

La définition de Copeland tente de réunir l'ensemble minimal de contraintes que doit satisfaire une formalisation pour être capable de calculer l'ensemble des fonctions calculables par MT. On remarque par exemple que le déterminisme de la procédure n'est pas une contrainte inscrite dans la définition puisqu'une MT déterministe calcule les mêmes fonctions qu'une MT non déterministe. De manière similaire, la contrainte stipulant que le calculateur n'a pas le droit de travailler en parallèle sur plusieurs rubans ne fait pas partie de cette définition car une MT travaillant sur n rubans avec $n > 1$ calcule les mêmes fonctions qu'une MT travaillant sur un seul ruban.

L'intérêt de cette définition est qu'elle permet d'étudier la puissance calculatoire des formalisations selon les contraintes qui sont satisfaites. En effet, si l'on suppose que les contraintes proposées par Copeland sont nécessaires et suffisantes pour caractériser la puissance calculatoire d'une MT - au sens où si une formalisation satisfait toutes et seulement ces contraintes alors elle calcule les mêmes fonctions que la MT - il suit qu'une formalisation ne satisfaisant pas une de ces contraintes devra nécessairement posséder une puissance inférieure ou supérieure à celle la MT. Autrement dit, cela signifie que la suppression d'une de ces contraintes modifie la puissance calculatoire de la formalisation fondée sur les contraintes restantes.

C'est justement en supprimant certaines contraintes énoncées dans la définition de Copeland qu'il est possible de formaliser des machines appelés *hyper-machines* (HM) ayant une puissance calculatoire supérieure à celle de la MT. Plus précisément, la puissance de calcul additionnelle d'une HM provient d'opérations fondamentales qu'elle peut exécuter contrairement à une MT. Cette puissance provient en particulier de la suppression de certaines contraintes sous-jacentes à la notion de procédure effective.

Dans le but d'illustrer de façon concrète l'origine de cette puissance de calcul, je présente dans ce qui suit certaines HM fondées explicitement sur la suppression de contraintes particulières. Mon but n'est pas de faire une liste exhaustive des différentes HM que l'on peut trouver dans la littérature [10]. Plutôt, je souhaite exhiber certaines HM dont le fonctionnement est clairement illustré par la suppression d'une contrainte énoncée dans la définition de procédure effective.

Machines de Turing à oracle. De la manière dont Turing l'a définie - à savoir comme une MT composée d'une boîte noire ou oracle capable de fournir les valeurs de fonctions non Turing-calculables, *cf.* introduction générale - une () n'est pas fondée explicitement sur la suppression d'une contrainte qui compose la notion de procédure effective [Turing, 1939, p.167]. Cependant, Copeland et Proudfoot ont proposé une autre définition de l'OM

10. Pour une présentation générale de ces HM, consulter [Copeland, 2002b] et [Stannett, 2004].

en supprimant la contrainte de finitude du nombre de symboles.

De leur point de vue, une OM est une MT composée de deux éléments : un dispositif capable d'exécuter des mesures d'une précision arbitraire et un espace mémoire qui contient une valeur précise appelée τ [Copeland and Proudfoot, 1999, p. 103]. Le nombre τ est un nombre représenté par une suite infinie de 0 et de 1 représentant les valeurs d'une fonction non Turing-calculable [11]. Si cette fonction est notée d, le n-ième symbole de τ représente $d(n)$ à savoir 0 ou 1. Et si l'on souhaite avoir accès au résultat $d(239208)$, le dispositif mesure le 239208-ième symbole de τ et en fournit la valeur. Par conséquent, une OM ayant en mémoire un nombre dont les décimales codent les résultats d'une fonction non Turing-calculable peut calculer davantage de fonctions que la MT.

Machines de Turing accélérantes. Dans leur ouvrage *Computability and Logic*, Boolos et Jeffrey imaginent une machine exécutée par Zeus - le principal dieu de l'ancien panthéon Grec - capable de résoudre certains problèmes indécidables par MT [Boolos and Jeffrey, 2005]. Plus précisément, Zeus a le pouvoir d'énumérer exhaustivement « devant ses yeux » n'importe quel *ensemble infini récursivement énumérable* [12]. Cette capacité lui permet en particulier de surpasser la MT car tout ensemble infini récursivement énumérable est dans ce cas considéré comme étant fini et donc devient trivialement décidable.

Plus récemment et d'après les travaux de Blake [Blake, 1926] et de Weyl [Weyl, 1927], Copeland a formalisé une autre HM en supprimant la contrainte de finitude du nombre d'étapes de calcul [Copeland, 2002c]. Cette HM nommée () a la possibilité de s'arrêter après un nombre infini d'étapes contrairement à

11. Les résultats d'une fonction de $\mathbb{N}$ dans $\mathbb{N}$ peuvent en effet être représentés par un nombre réel. La première définition que donne Turing du concept de calculable concerne justement les *nombres* calculables : « D'après ma définition, un nombre est calculable si ses décimales peuvent être inscrites par une machine » [Turing, 1936, p. 116]. Un nombre est ainsi calculable si une MT peut énumérer une à une ses décimales.

12. Informellement, un ensemble infini A est récursivement énumérable lorsqu'il existe une procédure qui fournit la réponse *oui* si un élément x - dont on souhaite déterminer s'il appartient à l'ensemble A - appartient à A, mais qui ne fournit aucune réponse si x n'appartient pas à A. L'ensemble A est récursif s'il existe une procédure qui fournit une réponse dans les deux cas.

une MT. Cette possibilité repose sur le principe suivant : le temps d'une étape de calcul est deux fois moins important que le temps de l'étape précédente. En particulier, si un calcul prend une unité de temps pour exécuter sa première itération, le temps total du calcul peut être exprimé par la série géométrique

$$\sum_{i=0}^{n} \frac{1}{2^i}$$

où i est la présente itération et n est le nombre d'itérations du calcul. Par conséquent, lorsque i tend vers l'infini, le temps total du calcul converge vers 2 car

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 2$$

Si le calcul peut être exécuté au moyen d'un nombre fini d'étapes, une MTA calcule exactement les mêmes fonctions que la MT. En revanche, si le calcul demande une infinité d'étapes pour arriver au résultat la MTA double sa vitesse à chaque étape et surpasse la puissance d'une MT.

L'avantage crucial d'une MTA par rapport à la MT repose sur le fait que la MTA s'arrête toujours. En effet, une MT qui ne s'arrête pas peut être interprétée de deux façons : soit elle s'arrêtera car elle va trouver un résultat, soit elle ne s'arrêtera jamais car il n'existe pas de résultat. Intuitivement, c'est l'impossibilité d'interpréter le non arrêt de la MT qui est la cause de la non Turing-calculabilité de certaines fonctions. En revanche, une MTA s'arrêtera toujours puisque qu'elle aura parcourue en un temps fini l'ensemble infini des étapes possibles de la MT.

Voici maintenant précisément pourquoi la MTA calcule davantage de fonctions que la MT - *cf.* Annexe A pour une définition formelle des classes de fonctions Turing-calculables.

Théorème

La classe Σ_1^0 est décidable par MTA.

Preuve : la classe Σ_1^0 est la classe des fonctions récursivement énumérables. Soit $f \in \Sigma_1^0$. D'une part, si $f \in \Sigma_1^0 \cap \Pi_1^0$ alors f est calculable par MTA car f est calculable par MT. D'autre part, montrons que si $f \in \Sigma_1^0 - \Sigma_1^0 \cap \Pi_1^0$

alors f est calculable par MTA. Soit la fonction récursivement énumérable g définie par $g(x) = 1$ s'il existe un entier naturel n tel que $g(n) = 0$ et $g(x) = 0$ autrement - g est une fonction unaire mais l'argument est valable quelque soit l'arité de la fonction. En premier lieu, g n'est pas calculable par une MT puisque dans le cas où g n'admet pas de solution le résultat de la MT diverge - la machine ne s'arrête jamais. En revanche, une MTA peut tester l'ensemble des entiers naturels en un temps fini à la recherche d'une solution à l'équation $g(n) = 0$. Dans le cas où cette dernière trouve un entier n satisfaisant l'équation, la tête de lecture de la MTA revient au début de son calcul et inscrit 1, autrement elle inscrit 0. La fonction g est donc calculable par MTA □

Machines de Turing à temps infini. La MTA n'est qu'un cas particulier d'une classe plus générale d'HM fondées sur l'exécution d'un nombre infini d'étapes de calcul.

Les HM exécutant une infinité d'étapes en un temps fini ont été généralisées par Hamkins et Lewis [Hamkins and Lewis, 2000], [Hamkins, 2002]. Ces derniers ont en particulier proposé une théorie mathématique décrivant les possibilités calculatoires des () qui sont des MTA pouvant réitérer - en un temps fini - l'exécution d'un nombre infini d'étapes de calcul.

Une MTI est une machine composée d'un dispositf qui est capable d'exécuter ω étapes entre deux unités de temps - ω représente le premier *ordinal infini* [Krivine, 2007]. La MTI peut faire débuter le calcul du dispositif et revenir à son état initial après deux unités de temps en laissant sur le ruban les calculs qui ont été inscrits entre ces deux unités de temps. La MTI peut alors faire débuter une seconde fois le dispositif pendant deux unités de temps. De cette manière la MTI est capable d'exécuter deux suites infinies d'étapes l'une après l'autre. Enfin, si la MTI est capable d'exécuter chaque suite deux fois plus vite que la précédente, cette dernière peut aussi exécuter une infinité de suites infinies successives.

Formellement, le nombre des étapes peut être représenté par les nombres

ordinaux :

$$0, 1, 3, \ldots, \omega, \omega + 1, \omega + 2, \ldots, \omega.2, \omega.2 + 1, \ldots, \omega^2, \omega^2 + 1, \ldots, \omega\omega, \ldots$$

D'après ce formalisme, le dispositif calcule ω étapes entre deux unités de temps et $\omega.2$ étapes s'il calcule une seconde fois. Enfin, la MTI exécute ω^2 étapes si elle est capable d'exécuter une infinité de suites infinies.

Machines à essais et erreurs. En 1965, Gold et Putman ont chacun publié un article sur le même sujet : *la récursivité limitative* [Gold, 1965], [Putnam, 1965]. Ce type de récursivité peut être réalisé sous la forme de (). Une MEE est une MT qui peut être utilisée pour déterminer si un élément x appartient à un ensemble $X \subseteq \mathbb{N}$ ou, plus généralement, si un n-tuple $(x_1, \ldots, x_n)$ appartient à une relation $R \subseteq \mathbb{N}^n$. Lors de son calcul, la MEE inscrit en continu une suite de réponses - de 0 et de 1 par exemple - où la dernière réponse est toujours la réponse correcte. Ainsi, si la réponse la plus récemment inscrite est 1, on sait que l'entier - ou le tuple - qui a été fourni en entrée doit appartenir à l'ensemble - ou à la relation.

De la manière dont elle est définie, la MEE n'est cependant pas capable d'hyper-calculer. En effet, la dernière réponse que la machine fournie peut être interprétée comme la réponse correcte *tant qu'elle* ne change pas d'état interne - c'est-à-dire tant qu'elle ne change pas le 1 en 0. Le problème est qu'il n'existe pas de procédure effective qui permet de déterminer si la MEE changera d'état interne ou non.

Supposons maintenant que la MEE puisse inscrire une infinité de réponses. D'après cette nouvelle opération, la machine convergera à partir d'un certain point vers une réponse particulière et incrira à partir de ce point toujours la même réponse - 0 ou 1. Autrement dit, la MEE inscrira une réponse correcte à partir d'un certain point, ce qui lui permettra de décider certains ensembles ou prédicats non décidables par MT. Pour comprendre ce point essentiel, précisons la notion de *prédicat récursif limitatif* :

Définition (Prédicat récursif limitatif)
Une fonction P est un prédicat récursif limitatif s'il existe une fonction

récursive f telle que pour tout $(x_1, \ldots, x_n)$,

$$P(x_1, \ldots, x_n) \longleftrightarrow \lim_{y \to \infty} f(x_1, \ldots, x_n, y) = 1,$$

$$\neg P(x_1, \ldots, x_n) \longleftrightarrow \lim_{y \to \infty} f(x_1, \ldots, x_n, y) = 0,$$

où

$$\lim_{y \to \infty} f(x_1, \ldots, x_n, y) = k = \exists y \ \forall z \ (z \leq y \rightarrow f(x_1, \ldots, x_n, z) = k)$$

A partir de cette définition, Putman a démontré le théorème suivant [Putnam, 1965] :

Théorème
P est un prédicat récursif limitatif si $P \in \Delta_2^0$.

Puisque la classe des fonctions Turing-calculables est la classe Δ_1^0 - classe ne pouvant pas être décidée par une MT - le théorème prouvé par Putnam énonce qu'une MEE peut dépasser la puissance calculatoire de la MT.

Machines manipulant des nombres réels. Les travaux de Turing ont pour objet la calculabilité des fonctions définies sur les entiers naturels. Toutefois, les fonctions utilisées en mathématiques sont généralement définies sur $\mathbb{R}$ - l'ensemble des nombres réels. Même si Turing a montré dans son article de 1936 que les fonctions définies sur $\mathbb{R}$ couramment utilisées en mathématiques sont calculables, il n'a pas proposé de théorie générale à propos de la calculabilité des fonctions sur $\mathbb{R}$.

Certains auteurs tels que Weirauch ont néanmoins tenté de définir la calculabilité sur les nombres réels [Weihrauch, 2000]. Cependant, la notion de calculabilité qui se dégage de ces travaux n'est pas aussi *complète* que celle établie sur les entiers naturels. Par exemple même si certaines formalisations définies sur $\mathbb{R}$ calculent les mêmes fonctions que la MT, il est très difficile d'établir une thèse analogue à la thèse de Church-Turing. En effet, il n'existe pour le moment aucune classe de formalisations assez générale pour définir les fonctions réelles calculables, au même titre que la classe des formalisations

sur $\mathbb{N}$ - MT, fonctions récursives, fonctions λ-calculables - qui permet de définir les fonctions calculables sur $\mathbb{N}$ [Bournez, 2008].

Une des raisons illustrant cette difficulté est que la frontière entre les formalisations sur $\mathbb{R}$ qui calculent de manière effective et les formalisations sur $\mathbb{R}$ qui hyper-calculent n'est pas clairement définie. Il est en particulier possible de formaliser des HM si l'on supprime la contrainte selon laquelle les données en entrée et en sortie doivent être des nombres entiers. Par exemple, les trois machines suivantes manipulent des nombres réels et sont capables de calculer davantage de fonction que la MT :

1. Les ont été introduites par Blum, Shub et Smale dans le but de décrire les calculs sur les nombres réels [Blum et al., 1926]. Même si ces dernières n'ont pas été conçues à proprement parler pour calculer des fonctions non Turing-calculables mais pour étudier les classes de complexité, elles peuvent néanmoins être considérées comme des HM.
2. Les ont été proposées par Copeland et sont capables de faire des opérations arithmétiques sur des données en entrée reélles [Copeland, 1997]. Les résultats de telles machines ne sont pas Turing-calculables.
3. Les proviennent des travaux d'Eilenberg [Eilenberg, 1974] mais ont été étudiées dans le cadre de l'hyper-calcul par Stannett [Stannett, 1990], [Stannett, 2001b]. Ces dernières sont en particulier capables de résoudre des problèmes indécidables par MT énoncés sous la forme de fonctions continues.

Bien qu'étant énoncées de façons différentes, ces trois HM peuvent calculer des fonctions non Turing-calculables à l'aide de nombres réels. Même si je ne vais pas rentrer dans les détails de ces formalisations, il est assez simple de comprendre pourquoi de telles machines peuvent calculer plus que la MT.

Tout d'abord, si la machine ne manipule que des nombres réels non Turing-calculables - qui sont des nombres dont les décimales sont les résultats de fonctions non Turing-calculables - alors il est clair que la machine est dans ce cas une HM. Ensuite, si la machine manipule des nombres réels *arbitraires* une simple considération sur la cardinalité montre que le nombre produit par la machine est non Turing-calculable avec une probabilité égale à 1 [Ca-

lude and Svozil, 2008]. Il existe en effet une quantité infinie non dénombrable de nombre réels tandis qu'il n'existe qu'une quantité infini dénombrable de fonctions Turing-calculables, à savoir de nombres pouvant être produits par une MT. Par conséquent en supposant que chaque nombre réel arbitraire à la même probabilité d'être produit en résultat, la probabilité que la machine produise un nombre réel Turing-calculable est égale à 0 ; tandis que la probabilité que le nombre soit non Turing-calculable est égale à 1.

Les formalisations des HM à l'aide d'opérations introduites *via* la suppression de certaines contraintes permet de résoudre le problème de l'opposition entre calcul effectif et hyper-calcul. Tout d'abord, il est à présent possible de reformuler ce problème en introduisant la notion de contrainte satisfaite par une machine : affirmer que l'hyper-calcul est en opposition avec la TCT revient à défendre la thèse selon laquelle certaines machines satisfaisant *toutes et seulement* les contraintes énoncées dans la définition de procédure effective peuvent calculer davantage de fonctions que la MT. J'ai cependant montré que la puissance supplémentaire octroyée aux HM provenait de la suppression de certaines contraintes telle que la finitude du nombre d'étapes de calcul. Par conséquent puisque les HM ne sont pas formalisées à partir des mêmes contraintes que la MT, l'hyper-calcul ne remet pas en cause la TCT.

Plus généralement, l'hyper-calcul et le calcul effectif peuvent être considérés comme des définitions alternatives de la notion de calcul. En ce sens, il n'existe pas *une* définition de la notion de calcul mais plusieurs définitions à partir desquelles on peut formaliser des machines possédant des puissances calculatoires différentes. Autrement dit,

> La calculabilité est une notion relative, et non une notion absolue. Tout calcul peut être effectué relativement à un ensemble de capacités primitives. Plus riches sont les capacités disponibles, plus grande est l'extension du calculable [Copeland, 2002b, p. 462].

De ce point de vue, plusieurs notions de calcul peuvent être définies en fonction des contraintes que l'on introduit dans leur définition. Ces contraintes permettent de caractériser de façon précise la puissance calculatoire des machines que l'on peut formaliser. Il n'est donc pas contradictoire que le calcul

effectif et l'hyper-calcul soient logiquement possibles.

L'opposition entre calcul effectif et hyper-calcul n'est cependant pas la seule objection adressée contre la possibilité logique de l'hyper-calcul. Il existe en effet une seconde objection qui est énoncée contre les processus utilisés par les HM pour hyper-calculer, à savoir l'utilisation de processus infinis.

2 Hyper-calcul et infini

Dans cette section, je combats une seconde objection énoncée contre de la possibilité logique de l'hyper-calcul. Contrairement à la première, cette objection n'est pas centrée sur la notion d'hyper-calcul mais sur les processus infinis qui sont utilisés par les HM pour hyper-calculer.

La formalisation des HM à partir de la suppression des contraintes inscrites dans la définition de procédure effective permet de mettre en lumière l'origine de leur puissance calculatoire : l'utilisation de processus infinis. Précisément, c'est l'introduction d'un élément ou d'un processus infini au sein d'une formalisation qui permet de passer de la MT à une HM, c'est-à-dire du calcul effectif à l'hyper-calcul.

Chaque contrainte que doit satisfaire la MT illustre en effet une caractéristique finie de la machine [13] :

1. Le nombre de ses étapes de calcul doit être fini.
2. L'espace mémoire qui peut être utilisé lors d'un calcul est fini.
3. Le nombre de ses transitions ou états computationnels doit être fini.
4. Sa tête de lecture ne peut reconnaître qu'un nombre fini de symboles.

Mais il est possible de formaliser une HM si l'on supprime la finitude d'une de ces contraintes :

1. Une machine de Turing accélérante peut exécuter une infinité d'étapes en un temps fini [Copeland, 2002a].

13. Ce point de vue est renforcé par les preuves de Gandy [Gandy, 1980] et de Sieg [Sieg, 2002] établissant que de nombreuses classes de machines différentes ne pouvant utiliser que des processus finis sont équivalentes à la MT.

2. Une machine de Turing à oracle garde en mémoire un nombre comportant une infinité de décimales [Copeland and Proudfoot, 1999].
3. Une machine de Turing à états infinis peut effectuer une infinité de transitions [Ord, 2002].
4. Les machines proposées par Boolos et Jeffrey peuvent énumérer en une seule étape de calcul l'ensemble infini des entiers naturels [Boolos and Jeffrey, 2005].

L'utilisation de processus infinis a suscité de nombreuses objections. Néanmoins, ce sont les HM ayant la capacité de calculer une infinité d'étapes en un temps fini qui ont été soumises à la majorité des critiques. D'une part, le concept d'une infinité d'étapes exécutée en un temps fini - appelé dans la littérature *supertask* - est au centre de nombreuses controverses liées aux paradoxes logiques de l'infini. D'autre part et de manière plus technique, certains auteurs défendent en s'appuyant sur les travaux de Turing que les HM pouvant exécuter un supertask sont logiquement contradictoires car elle peuvent calculer leurs propres fonctions arrêt et diagonale. Turing a en effet montré dans ses travaux de 1936 et de 1939 que ni les MT ni les OM ne sont capables de calculer leurs propres fonctions arrêt et diagonale sans être contradictoires.

Dans ce qui suit, je vais tenter de rejeter ces deux objections. Je commence dans un premier temps par soutenir qu'il est impossible de déduire des paradoxes logiques de l'infini que l'exécution d'un *supertask* est contradictoire. J'explique dans un second temps pourquoi les HM fondées sur l'exécution d'un supertask ne sont pas capables de calculer leurs propres fonctions arrêt et diagonale.

2.1 Les paradoxes logiques de l'infini

L'exécution d'un nombre infini d'étapes en un temps fini - processus appelé () - a soulevé de nombreuses objections. En philosophie, ces objections prennent généralement la forme de paradoxes servant à montrer qu'il est logiquement impossible d'exécuter un ST. Voici les trois paradoxes logiques qui ont été le plus souvent invoqués :

1. Les paradoxes de Zénon [Huggett, 2010].
2. Les paradoxes de Thomson [Thomson, 1954].
3. Le paradoxe de Ross [Ross, 1988].

Mon but est ici de montrer que ces trois paradoxes logiques de l'infini sont insuffisants pour conclure que les ST sont logiquement contradictoires. Je commence par les paradoxes de Zénon.

2.1.1 Les paradoxes de Zénon

Les paradoxes logiques de l'infini ont traditionnellement pour origine les paradoxes de Zénon liés à la possibilité du mouvement [Huggett, 2010]. Les paradoxes du mouvement sont au nombre de quatre :

1. **La dichotomie** : il est impossible d'atteindre un point situé à une distance donnée car il est à chaque fois nécessaire d'atteindre au préalable le point situé à mi-chemin de cette distance.
2. **Achille et la tortue** : Achille, bien plus rapide que la tortue avec laquelle il fait une course, ne peut cependant jamais la dépasser s'il lui laisse une quelconque avance initiale car il lui reste toujours une distance à parcourir avant de la dépasser.
3. **La flèche** : une flèche lancée est toujours immobile. En effet, tout corps est soit en mouvement soit en repos quand elle se trouve dans un espace égal à son volume ; or la flèche se trouve à chaque instant dans un espace égal à son volume.
4. **Le stade** : la moitié du temps est égale au double. Zénon arrive à cette conclusion en raisonnant à partir d'un stade dans lequel se déplacent en sens inverse des rangées de spectateurs.

Les deux paradoxes de Zénon qui ont été le plus débattus sont incontestablement la dichotomie et la course entre Achille et la tortue. Dans ce qui suit, je ne vais traiter que du paradoxe de la dichotomie pour une raison bien précise : la modélisation de l'expérience de pensée qui implique le paradoxe de la dichotomie est similaire à la modélisation du fonctionnement de la machine

de Turing accélérante, HM capable d'exécuter un ST pour hyper-calculer [14].

Commençons par présenter le paradoxe de la dichotomie. Selon ce paradoxe, Achille est initialement au point A ($x = 0$) et veut atteindre d'une ligne droite un point B ($x = 1$). Pour Zénon, Achille n'atteindra toutefois jamais le point B ($x = 1$). En effet, le coureur Grec devra d'abord atteindre le point $x = \frac{1}{2}$ au milieu des points A et B. Il devra ensuite atteindre le point $x = \frac{1}{2} + \frac{1}{4}$ qui est le milieu des points $x = \frac{1}{2}$ et B. En continuant ainsi le raisonnement, il sera impossible pour Achille d'atteindre sa destination car la distance entre A et B prendra la forme d'une somme infinie. Autrement dit, à chaque fois qu'Achille aura atteint un point il existera toujours un nouveau point qu'il n'aura pas atteint.

Pour Zénon, la dichotomie conduit à un paradoxe logique qui a deux conséquences pour l'hyper-calcul :

1. Les ST sont logiquement contradictoires.
2. Les MTA sont logiquement contradictoires.

Il est en effet possible de reformuler la course d'Achille comme un ST. Plus précisément, Achille doit exécuter une infinité d'étapes en un temps fini s'il veut atteindre le point B. Si l'on suppose qu'Achille commence sa course au temps $t = 12$ h, que le point B est situé à 1 km du point A et qu'il se déplace à une vitesse constante $v = 1$ km/h, Achille devrait atteindre le point B au temps $t^* = 13$ h. Toutefois, en reprenant le raisonnement de Zénon, lorsque la moitié du temps nécessaire pour atteindre $t^* = 13$ h sera écoulée, Achille exécutera l'action a_1 et aura bougé du point A ($x = 0$) au point A_1 ($x = \frac{1}{2}$). Lorsque la moitié du temps entre l'action a_1 et $t^* = 13$ h sera écoulée, Achille exécutera l'action a_2 et sera passé du point A_1 ($x = \frac{1}{2}$) au point A_2 ($x = \frac{1}{2} + \frac{1}{4}$). En réitérant ce processus, on peut conclure que lorsque le point B sera atteint au temps $t^* = 13$ h, Achille aura exécuté une série infinie d'actions $S = (a_1, a_2, a_3, \ldots, a_n, \ldots)$.

Le fonctionnement d'une MTA repose en outre sur l'exécution d'un ST. La MTA est en effet capable de doubler sa vitesse à chaque étape de calcul ce

14. Pour une étude du paradoxe de la course d'Achille et la tortue, consulter en particulier [McLaughlin, 1994], [Salmon, 2001] et [Ardourel, 2013] (chapitre 6).

qui implique une réduction progressive de l'intervalle de temps entre chaque étape, processus pouvant être modélisé par une série géométrique infinies. Par conséquent si le raisonnement de Zénon s'avère être juste, les concepts de ST et de MTA seraient logiquement contradictoire.

Il existe néanmoins une résolution standard du problème zénonien. Cette solution consiste à considérer la somme infinie S comme une série mathématique. La théorie des séries mathématiques permet de calculer la limite de la somme infinie $S = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots$ et d'en déduire qu'elle est finie et égale à 1. Contrairement au paradoxe de la course d'Achille qui repose sur l'hypothèse selon laquelle une somme infinie est infinie, la théorie des séries permet de conclure qu'Achille atteindra bien le point B en parcourant une distance finie. Il n'existe alors aucun problème apparent concernant le fait de diviser la course en une infinité de sous-courses. En résumé, la résolution standard est en faveur de l'hyper-calcul.

Même si elle n'est actuellement pas rejetée, la résolution standard soulève certains doutes quant à la facilité avec laquelle elle parvient à résoudre le paradoxe de zénon.

Il est tout d'abord pertinent de reprocher à la résolution standard que même si elle permet de surmonter le paradoxe, cette dernière n'explique pas en quoi l'argument de Zénon faillit. Pour certains, cette solution pourrait même être considérée comme une pétition de principe puisque calculer mathématiquement la somme infinie de la série S semble présupposer que l'on puisse effectuer un nombre infini d'opérations en un temps fini ; alors même que ce qui est recherché est la démonstration de cette possibilité [Chihara, 1965].

De plus, la résolution standard ne permet pas de résoudre une seconde forme du paradoxe de Zénon appelée *dichotomie inverse* qui peut sembler encore plus problématique que la première. Le processus issu de la dichotomie inverse conduit au ST suivant. Supposons qu'Achille soit au point A ($x = 0$) à $t = 12$ h et qu'il veuille entreprendre une action dans le but d'atteindre le point B ($x = 1$). Pour ce faire, il devra d'abord atteindre le point B_1 ($x = \frac{1}{2}$) qui est le milieu de la distance AB. Ensuite, Achille devra rejoindre le point B_2 ($x = \frac{1}{4}$) milieu de AB_1. En suivant cette logique, Achille devra accomplir une

infinité d'actions représentée par $T^* = (\ldots, a_n, \ldots, a_3, a_2, a_1)$. Le problème est ici que la relation d'ordre sur le domaine de T^* est la même que la relation d'ordre sur l'ensemble des entiers négatifs $(\ldots, -n, \ldots, -3, -2, -1)$. Plus précisément, un des problèmes majeures liés à T^* découle de l'impossibilité de décrire l'état initial du processus. En effet, puisque les états du processus décrits par T^* peuvent être représentés par un ensemble infini d'énoncés, on ne peut ni décrire son état initial, ni spécifier sa première action. Par conséquent puisqu'Achille est supposé pouvoir énumérer les entiers naturels dans un ordre inversé, ce dernier ne peut pas commencer à courir.

Il est enfin légitime de reprocher aux défenseurs de la résolution standard de déduire des énoncés empiriques à partir de définitions sur les séries arithmétiques infinies. La résolution du paradoxe ne devrait donc pas provenir de considérations mathématiques puisque le paradoxe en lui-même est un problème physique [Morris, 1997], [Lynds, 2003].

Si la résolution standard du paradoxe de Zénon ne fait pas l'unanimité, les objections énoncées à son encontre ne démontrent pas pour autant que le concept de ST est logiquement contradictoire - au mieux illustrent-elles la position selon laquelle l'exécution d'un ST ne doit pas se fonder sur la course d'Achille. Le paradoxe suivant reprend la position de Zénon sous une forme plus contemporaine qui a introduit au 20^{e} siècle le débat portant sur la possibilité des ST.

2.1.2 Les paradoxes de Thomson

James Thomson défend en 1954 la thèse selon laquelle l'exécution d'un ST est contradictoire [Thomson, 1954]. La question de la possibilité des ST a fait l'objet d'une discussion célèbre entre Thomson et Benacerraf dans les années cinquante. Tandis que Thomson tentait de prouver que les ST sont impossibles, Benacerraf s'efforçait à montrer que les arguments de Thomson n'étaient pas pertinents [Benacerraf, 1962]. Dans ce qui suit, je retraçe les principaux arguments de cette controverse et j'explique pourquoi la position de Benacerraf me semble être la plus pertinente.

Afin de mieux comprendre le point de vue de Thomson, commençons par

résumer le raisonnement de Zénon qui conduit selon lui à la contradiction logique de la notion de ST :

Première prémisse Pour parcourir une distance, il est nécessaire de parcourir une infinité de distances.

Seconde prémisse Il est absurde que quelqu'un puisse parcourir entièrement une infinité de distances.

Conclusion Il est absurde que quelqu'un puisse parcourir une quelconque distance.

Il existe en particulier deux analyses différentes des prémisses utilisées par Zénon qui conduisent toutes deux à la validité du raisonnement. La différence entre ces deux analyses réside dans l'interprétation de l'expression *parcourir un nombre infini de distances*.

D'une part, si l'on soutient que parcourir une infinité de distances revient à parcourir une distance continue d'un point A vers un point B, alors parcourir la distance AB revient à traverser toutes les parties de AB comprenant AA' - où A' est le milieu de AB - A'A" - où A" est le milieu de AA' - et ainsi de suite. Autrement dit, il est impossible rejoindre le point B s'il existe au moins une partie de AB qui n'a pas été parcourue. D'après cette interprétation, la première prémisse est vraie tandis que la seconde est fausse car parcourir une infinité de distances revient dans ce cas à parcourir une unique distance - ce qui est possible d'après la validité de la première prémisse.

D'autre part, on peut interpréter l'expression *parcourir un nombre infini de distances* comme le fait d'exécuter une infinité d'actes physiques distincts. Dans ce cas la première prémisse est évidemment fausse puisqu'elle est contredite par l'expérience au quotidien. La seconde prémisse est en revanche vraie car on ne peut pas exécuter entièrement une infinité d'actes physiques distincts.

Contrairement à Zénon qui énonce son paradoxe à partir de la première interprétation, Thomson considère en revanche la seconde interprétation pour montrer qu'il est impossible d'exécuter une infinité d'actes physiques dis-

tincts[15]. Dans ce but, il énonce son premier argument sous la forme d'une expérience de pensée appelée *la lampe de Thomson*.

1^{er} **argument** : Considérons une lampe pouvant être soit allumée soit éteinte à partir d'un bouton. Lorsque le bouton est en position *on* la lampe est allumée ; lorsque le bouton est en position *off* elle est éteinte. Supposons ensuite que la lampe est éteinte et que la position du bouton est modifiée une infinité de fois en appuyant la première fois après 1 seconde, puis la seconde fois après $\frac{1}{2}$ seconde, *etc.* Quel est l'état de la lampe après une infinité de positions *on/off* ? Pour Thomson, la lampe ne peut pas être éteinte à la fin de l'expérience puisque chaque fois que le bouton est en position *off*, l'action suivante la remet sur *on*. Autrement dit, la lampe ne peut jamais être éteinte sans être rallumée. De même, la lampe ne peut pas être allumée à la fin de l'expérience puisque chaque fois que le bouton est en position *on*, l'action suivante la remet sur *off*. Autrement dit, la lampe ne peut jamais être allumée sans être éteinte. Pourtant, la lampe doit forcément être soit allumée soit éteinte lorsque l'expérience se termine. Puisque l'état de la lampe à la fin de l'expérience ne peut pas être déterminée, Thomson conclue que l'exécution d'une infinité d'actions en un temps fini est logiquement impossible.

Une possible solution au paradoxe de Thomson a été apportée par Paul Benacerraf en 1962. Pour montrer que la lampe de Thomson ne montre pas que l'exécution d'un ST est contradictoire, Benacerraf raisonne de la façon suivante. Si on appelle t_0 le moment où commence l'expérience et t_1 le moment où elle se termine, l'argument de Thomson indique qu'à t_1 la lampe n'est ni allumée ni éteinte. Benacerraf répond que la description de Thomson, même si elle permet de définir l'état de la lampe au cours de l'intervalle $[t_0, t_1[$ - une alternance de position *on/off* - ne permet pas de définir l'état de la lampe au temps t_1 : c'est-à-dire à la fin de l'expérience. Plus précisément, puisque l'alternance des positions *on/off* n'est vraie que pour les temps antérieurs à t_1, les raisons poussant Thomson à affirmer que la lampe est contradictoire ne sont valables que pour les instants avant t_1. Autrement dit, l'état de la lampe à t_1 n'est pas une conséquence logique de l'état

15. C'est pourquoi on surnomme généralement les défenseurs de la position de Thomson, *les éléatiques modernes*.

de la lampe aux instants antérieurs à t_1 et Thomson ne peut pas logiquement conclure que l'expérience est impossible.

Suite à la réponse de Benacerraf, Thomson énonce un nouvel argument qui repose davantage sur les mathématiques [Benacerraf, 1962, p. 769].

2nd **argument** : supposons qu'il existe deux types de nombres, les *nombres réguliers* et les *nombres irréguliers*, de façon à ce que chaque nombre fasse partie d'un des deux types. Considérons à présent la série convergeante $E = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} \ldots$ dont le premier membre est irrégulier, le second régulier, le troisième irrégulier, *etc.* Plus précisément, $\frac{1}{2^n}$ est irrégulier si n est pair et régulier si n est impair. Mais qu'en est-il de la limite de la série ? Est-elle représentée par un nombre régulier ou par un nombre irrégulier ? Ces questions sont pertinentes car même si la limite ne fait pas partie de E, elle doit néanmoins être par définition un nombre régulier ou irrégulier. Le problème est que la limite ne peut pas être un nombre régulier puisque dans E chaque nombre régulier est suivi par un nombre irrégulier ; elle ne peut pas être non plus un nombre irrégulier pour la même raison. Puisque tout nombre est soit régulier soit irrégulier le fait que la limite ne puisse pas être définie comme étant un nombre régulier ou un nombre irrégulier amène ainsi à la conclusion selon laquelle la notion de ST est contradictoire.

Pour résoudre le nouveau paradoxe de Thomson, Benacerraf utilise le même type d'argument que le précédent. La description des nombres réguliers et irréguliers n'apporte aucun renseignement à propos de la classification du nombre limite, elle se contente de classer les nombres de la série. En ce sens, aucune réponse à la question « la limite est-elle un nombre régulier ou irrégulier ? » ne peut être apportée à cause d'un manque d'informations dans l'énoncé du problème.

Thomson énonce enfin un troisième argument qui semble convaincant au premier abord mais qui se révèle une fois de plus défectueux.

3e **argument** : Supposons que la lampe puisse prendre deux valeurs : la valeur 1 et la valeur 0. Supposons ensuite que l'action d'allumer la lampe ajoute 1 et que l'action d'éteindre la lampe soustraie 1. La question de savoir si la lampe est allumée ou éteinte après avoir modifié une infinité de fois la position du bouton revient donc à déterminer la valeur de la lampe après

une alternance infinie d'additions et de soustractions du chiffre 1. On peut en particulier représenter cette expérience par la suite [16] $+1, -1, +1, -1, \ldots$ qui a une somme $S = \frac{1}{2}$. Cependant, il n'y a aucun sens à dire que la lampe est en position $\frac{1}{2}$ c'est-à-dire entre la position allumée et la position éteinte.

Ce troisième argument a pour caractéristique de ne soulever aucune contradiction au niveau mathématique. La contradiction réside au niveau physique puisque qu'il n'existe pas dans notre monde d'analogue physique à l'état $\frac{1}{2}$, où une lampe est à la fois allumée et éteinte. L'argument de Thomson semble par conséquent pertinent relativement à une interprétation physique de la possibilité d'un ST.

Ici encore, Benacerraf montre qu'il n'en est rien. Pour cela, ce dernier commence par rappeler que pour chaque entier positif n, la valeur de la lampe après avoir modifié n fois la position du bouton est représentée par sa somme partielle - notée SP_n. Benacerraf pose ensuite la question suivante : quelles raisons avons-nous de penser que la valeur de la lampe après avoir modifié n fois la position du bouton sera représentée par la somme de tous ses termes, c'est-à-dire par la limite de la somme des SP_n qui est égale à $\frac{1}{2}$? Aucune nous dit-il, car le fait que $\frac{1}{2}$ ne puisse représenter l'état de la lampe ne montre pas qu'il ne peut pas exister une infinité d'actions. L'argument de Thomson définit uniquement l'état de la lampe pour chaque SP_n ce qui n'implique pas une définition de l'état de la lampe après le ST. Pour résumer, il n'y a aucune raison de croire que parce que les SP_n ont une certaine propriété - représenter l'état de la lampe - la somme des SP_n doit aussi posséder cette propriété.

Bien qu'il existe d'autres raffinements au sein de la controverse Thomson/Benacerraf, l'essentiel a été d'en avoir examiné les principaux arguments. Plus précisément, l'analyse de ces arguments a permis de comprendre comment la position de Benacerraf permet de contrer les arguments de Thomson énoncés contre la possibilité logique des ST.

16. Cette suite appelée *série de Grandi* est la série associée à la suite $u_n = (-1)^n$ dont les sommes partielles sont $1-1+1-1+\ldots+(-1)^{n-1}$. Bien que cette série soit divergente, la *moyenne de Cesàro* - qui est obtenue en effectuant la moyenne arithmétique des n premiers termes de la suite - des sommes partielles converge vers $\frac{1}{2}$. On associe donc généralement à la série de Grandi la somme $S = \frac{1}{2}$.

L'étude des paradoxes de Zénon et de Thomson a permis de montrer que les premiers paradoxes énoncés à l'encontre des ST pouvaient être résolus ou du moins surmontés. Il existe cependant un troisième paradoxe bien plus problématique que les deux précédents. Ce paradoxe a été imaginé par Ross [Ross, 1988, p. 46] et Littlewood [Littlewood, 1986] et mérite d'être étudié pour deux raisons. D'une part, tandis que l'existence d'une limite mathématique permet de surmonter le paradoxe de Zénon, c'est justement l'existence d'une telle limite qui pose problème dans le pradoxe de Ross. D'autre part, les résolutions relatives au paradoxe de Ross sont si variées qu'il n'existe à ce jour aucune solution standard admise par tous.

2.1.3 Le paradoxe de Ross

Imaginons une urne disposant d'un espace infini et d'une infinité de boules numérotées $1, 2, 3 \ldots$ Maintenant, 1 minute avant minuit, les boules 1 à 10 sont insérées dans l'urne et la boule numéro 1 est retirée. A $\frac{1}{2}$ minute avant minuit, les boules 11 à 20 sont insérées et la boule numéro 2 est retirée, *etc.* Quel est alors le nombre de boules dans l'urne à minuit ?

Il semble d'une part que le nombre de boules à minuit soit infini puisque l'on a effectué une infinité d'opérations, chacune ajoutant un total de 9 boules dans l'urne. Il semble donc que le nombre N de boules à minuit soit

$$N = \lim_{n \to \infty} 9n = \infty$$

Mais puisque la boule n est enlevée à la n-ième opération et que n prend les valeurs de tous les entiers naturels, il semble d'autre part que l'urne soit vide à minuit.

Quelle est donc la réponse correcte ? Sur cette question, les points de vue de la communauté scientifique sont très variés mais Ross soutient davantage la seconde hypothèse car chaque boule n est enlevée à la n-ième étape. Pour évaluer si Ross a raison ou tort, j'explique dans ce qui suit les différentes approches proposées dans le but de répondre à la question soulevée par son expérience.

En premier lieu, la position soutenue par Ross selon laquelle l'urne doit

être vide à minuit est défendue par Allis et Koetsier [Allis and Koetsier, 1991]. Selon eux, la résolution de ce paradoxe implique *une interprétation cinématique* et *un principe de continuité* pour déterminer l'issue de l'expérience à minuit. Cela signifie que c'est à la physique d'apporter une solution au paradoxe de Ross et que les outils logiques ne sont pas suffisants pour y parvenir. Plus précisément, Allis et Koetsier interprètent l'expérience comme un ST exécuté dans un espace euclidien contenant plusieurs voir une infinité d'éléments rigides. Lors de l'expérience, les boules numérotées sont des éléments rigides qui se déplacent dans le plan et qui sont considérées comme des disques circulaires possédant une position précise dans ce même plan. L'urne est considérée comme un ensemble de points non vide dans le plan. Une boule est dite à l'intérieur de l'urne si le centre du disque circulaire qui la représente est un élément de cet ensemble de points. Elle est à l'extérieur de l'urne autrement.

Au cours de l'exécution du ST, les différentes boules sont déplacées. Le principe de continuité sur lequel est fondé leur raisonnement peut être énoncé de la façon suivante : si à un moment avant minuit, une boule est à une certaine position et qu'elle ne quitte pas cette position jusqu'à minuit, alors cette boule est encore à cette position à minuit. Autrement dit, le principe d'Allis et Koetsier énonce que si le vecteur (p, q) à un certain moment avant minuit devient un vecteur constant et le reste jusqu'à minuit, alors l'élément impliqué est encore à la même position à minuit [17].

Il est maintenant aisé de conclure que l'urne est vide à la fin de l'expérience. Pour arriver à cette conclusion, les boules doivent être positionnées à l'extérieur de l'urne à minuit. D'une part, lors de l'expérience chaque boule dont le numéro est n est mise à l'extérieur de l'urne avant minuit, avec $n \in \mathbb{N}$. D'autre part, puisque les boules ne changent pas de position avant minuit le

17. Les auteurs ont tout de même tenté de définir un principe de continuité qui ne serait pas fondé sur une interprétation cinématique. Ces derniers ont en particulier proposé le principe suivant : si à un certain moment avant minuit, la valeur de vérité d'un énoncé exprimant un fait lié à l'expérience devient et reste constant, alors la valeur de vérité de cet énoncé est identique à minuit. Ils se sont cependant vite rendu compte des problèmes que peut poser une telle définition. En effet, certains énoncés tels que *maintenant il n'est pas encore minuit* ou encore *il y a au moins une boule dans l'urne* ne doivent pas être pris en compte sous peine de mener l'expérience de Ross à une contradiction.

principe de continuité permet de conclure que les boules sont à l'extérieur de l'urne à minuit et donc que l'urne est vide.

Même si Allis et Koetsier parviennent à démontrer que l'urne est vide à minuit, leur résultat repose néanmoins sur une interprétation physique de l'expérience et plus particulièrement sur l'hypothèse de continuité. C'est pourquoi van Bendegem a proposé un argument qui est dénué de tout principe de continuité [Van Bendegem, 1994]. Cet argument, loin de confirmer le résultat d'Allis et Koetsier, en exhibe une contradiction.

D'après l'énoncé du paradoxe, $10 - 1$ boules sont en effet ajoutées dans l'urne à chaque étape du ST. La somme totale S de boules dans l'urne à la fin du ST est ainsi

$$S = (10 - 1) + (10 - 1) + \dots$$

Si l'on suppose que S est un nombre fini, à savoir $S = 0$ ou $S = n$ pour $n \in \mathbb{N}^*$, une contradiction peut être dérivée de la précédente formule - à savoir que $9 = 0$.

$$\begin{aligned} 9 + S &= (10 - 1) + S \\ &= (10 - 1) + (10 - 1) + (10 - 1) + \dots \\ &= S \end{aligned} \tag{1.1}$$

Cette dernière contradiction montre que l'urne ne peut pas être vide à minuit puisque si $S = 0$, $9 = 0$.

La conclusion de van Bendegem est tout à fait correcte et semble soutenir *via* une preuve mathématique que la solution d'Allis et Koetsier n'est pas valide. Toutefois, ces deux derniers auteurs ont tenté de contrer l'argument de van Bendegem en expliquant que cet argument n'est pas dénué de principe de continuité contrairement à ce qu'avait défendu van Bendegem :

> Notez que l'argument est entièrement indépendant de tout principe de continuité ou de toute supposition cinématique [Van Bendegem, 1994, p. 743].

Pour Allis et Koetsier, écrire $S = (10 - 1) + (10 - 1) + \dots$ comme le fait

van Bendegem signifie par définition que

$$S = \lim_{x \to \infty} (10 - 1)n$$

Cela implique (1) que $S = \infty$ et (2) qu'il existe une hypothèse de continuité à partir de laquelle il n'est pas difficile de déduire une contradiction si S est finie. Ainsi,

> [...] nous devons tirer la conclusion que la tentative de van Bendegem de montrer l'impossibilité de l'exécution du super-task de Littlewood et de Ross par des moyens logico-mathématiques échoue [Allis and Koetsier, 1995, p. 245].

Même si Allis et Koetsier mettent à mal l'argumentation de van Bendegem, ce dernier n'est pas le seul à défendre la thèse selon laquelle l'urne est remplie d'une infinité de boules à minuit. Selon Byl, il existe en effet une difficulté qui concerne la conclusion d'une urne vide à minuit :

> Quelle est la cause de l'évaporation soudaine des boules à minuit ? Que l'urne soit vide à minuit semble étonnant, parce que le nombre de boules dans l'urne continue d'augmenter jusqu'à minuit [Byl, 2000, p. 44].

Puisque le ST est défini comme un nombre infini d'actions similaires - chaque action consistant à ajouter 10 boules et à en enlever une - comment est-il possible que l'urne soit vide à minuit alors qu'à chaque action un total de 9 boules est inséré dans l'urne ? En d'autres termes, comment un ensemble infini d'additions peut-il être entièrement supprimé si aucune action n'implique une soustraction de cet ensemble ?

Une possible réponse est de dire que la soudaine évaporation des boules doit provenir d'une opération non définie exécutée à minuit [Earman and Norton, 1996]. L'argument selon lequel l'urne est vide à minuit doit par conséquent reposer sur une hypothèse supplémentaire qui montre que les boules sortent de l'urne avant minuit.

Byl répond à ce problème en expliquant pourquoi l'urne ne doit pas être vide à minuit mais doit au contraire contenir une infinité de boules [Byl, 2000]. Pour ce dernier, le ST est complètement achevé lorsque la dernière

boule est entrée dans l'urne. Plus précisément, puisqu'à chaque étape la boule n est enlevée et les boules $n+1$ à $n+10$ sont ajoutées, Byl conclue qu'à minuit les boules 1 à ω ont été enlevées mais que l'urne contient encore les boules $\omega+1$ à 10ω. De cette manière, même si un nombre infini de boules a été enlevé à minuit il reste encore une infinité de boules à l'intérieur de l'urne.

Aucune des réponses proposées afin de résoudre le paradoxe de Ross ne fait actuellement l'unanimité. Ce paradoxe, contrairement à ceux de Zénon ou de Thomson, pose donc de réels problèmes à propos de la possibilité logique des ST. En d'autres termes, le concept de ST ne pourra pas être considéré comme logiquement possible tant que le paradoxe de Ross ne sera pas résolu.

Cette dernière phrase n'est pas totalement vraie car je pense qu'il existe une autre solution afin de montrer que les ST sont logiquement possibles et qui ne passe pas par le paradoxe de Ross. Cette solution consiste à prouver que la notion de ST est *physiquement possible*. J'ai en effet expliqué dans l'introduction de ce chapitre que la possibilité physique est une condition suffisante à la possibilité logique. Une preuve de la possibilité physique d'un ST devient par conséquent la preuve de sa possibilité logique. Cependant, la question de la possibilité physique des ST - et plus généralement celle de la possibilité physique de l'hyper-calcul - ne sera pas traitée tout de suite mais au chapitre 3. Pour l'instant, je vais me concentrer sur le second type d'objections qui est énoncé contre les HM dont le fonctionnement est fondé sur un ST.

2.2 Contradiction par diagonalisation

Dans ce qui suit, je vais tenter de résoudre un problème technique énoncé contre la possibilité de calculer des fonctions non Turing-calculables. Ce problème est différent des problèmes traditionnels énoncés contre les HM. Généralement, pour montrer qu'une HM n'est pas capable d'hyper-calculer on essaie d'exhiber un problème interne à son fonctionnement. Il peut par exemple être reproché à une OM le fait que son fonctionnement soit une pétition de principe : on suppose que la machine dispose d'un oracle pou-

vant résoudre un problème indécidable et on en déduit qu'elle est capable de calculer la fonction associée à ce problème. La situation est en revanche ici très différente : on tente de montrer qu'une HM est capable de calculer une fonction associée à un problème et on en déduit une contradiction logique. Ce n'est donc pas *comment* l'HM calcule qui est au cœur du problème mais *qu'est-ce qu'*elle calcule.

Le problème que je vais traité concerne plus précisément les HM utilisant des processus infinis exécutés en un temps fini, à savoir les MTA (machines de Turing accélérantes), les MTI (machines à temps infinis) et les MEE (machines à essais et erreurs). Je ne m'occupe toutefois que du cas des MTA afin d'éviter principalement certaines redondances - les solutions proposées n'étant que très légèrement différentes en fonction de chaque type d'HM. Commençons tout d'abord par définir le problème en question.

2.2.1 Définition du problème

Le problème a été introduit par Svozil et Cotogno [Svozil, 1998], [Cotogno, 2003]. Ces derniers défendent dans leurs articles que les MTA sont logiquement contradictoires car la puissance calculatoire qui leur est octroyée est trop grande. Leur puissance est trop grande car elle leur permettrait de calculer certaines fonctions qui devraient être en théorie non calculables sous peine de contradiction logique. Ces fonctions sont au nombre de deux : *la fonction diagonale propre aux MTA* et *la fonction arrêt propre aux MTA*. Afin de mieux comprendre l'argument de Svozil et Cotogno, il est intéressant de mettre ces deux fonctions en parallèle avec la fonction arrêt propre aux MT et la fonction diagonale propre aux MT. Commençons par définir les deux dernières fonctions.

Définition (Fonctions arrêt et diagonale propres aux MT)
Soient $MT_1, MT_2, MT_3 \ldots$ une énumération des MT - cette énumération est dénombrable. Soient H_{MT} la fonction arrêt propre aux MT définie par

$$H_{MT}(x,y) = \begin{cases} 0 \text{ } si\ MT_x(y)\ ne\ s'arrête\ pas\ après\ un\ nombre\ fini\ d'étapes, \\ 1 \text{ } autrement. \end{cases}$$

et D_{MT} la fonction diagonale propre aux MT définie par

$$D_{MT}(x) = \begin{cases} 0 \text{ si } H_{MT}(x,x) = 0 \\ \text{autrement diverge.} \end{cases}$$

La preuve classique montrant qu'une MT qui aurait la possibilité de calculer sa propre fonction arrêt serait contradictoire a été apportée par Turing [Turing, 1936]. Celle montrant que la fonction diagonale propre aux MT n'est pas Turing-calculable peut être énoncée comme ceci. Montrons que D_{MT} n'appartient pas à l'ensemble des fonctions Turing-calculables. Si D_{MT} est Turing-calculable, $\exists i \ / \ MT_i(x) = D_{MT}(x)$. De plus, cela implique que $MT_i(i) = 0 \iff D_{MT}(i) = 0 \iff H_{MT}(i,i) = 0 \iff MT_i(i)$ diverge, ce qui est contradictoire. Ainsi $D_{MT}(x)$ n'est pas Turing-calculable.

Continuons avec les fonctions arrêt et diagonale propres aux MTA.

Définition (Fonctions arrêt et diagonale propres aux MTA) *Soient $MTA_1, MTA_2, MTA_3 \ldots$ une énumération des MTA - cette énumération est aussi dénombrable puisqu'une MTA est définie de la même façon qu'une MT. Soient H_{MTA} la fonction arrêt propre aux MTA définie par*

$$H_{MTA}(x,y) = \begin{cases} 0 \text{ si } MTA_x(y) \text{ ne s'arrête pas après un nombre fini d'étapes,} \\ 1 \text{ autrement.} \end{cases}$$

et D_{MTA} la fonction diagonale propre aux MTA définie par

$$D_{MTA}(x) = \begin{cases} 0 \text{ si } H_{MTA}(x,x) = 0 \\ \text{autrement diverge.} \end{cases}$$

Ici encore, Turing a prouvé en 1939 que les OM ne peuvent calculer leur propre fonction arrêt sans entraîner de contradiction [Turing, 1939]. Par ailleurs, en suivant le même raisonnement que celui concernant les MT on peut montrer que la fonction diagonale propre aux MTA n'est pas calculable par une MTA.

Le problème de Cotogno et Svozil peut maintenant être énoncé sous la forme suivante : puisque Turing a montré que ni les MT ni les OM ne peuvent

calculer leurs propres fonctions arrêt et diagonale, si l'on prouve que les MTA sont assez puissantes pour calculer l'une de ces deux fonctions, elles se révèleraient de fait contradictoires.

Dans le but de résoudre ce problème, je vais procéder de la manière suivante. Je vais tout d'abord expliquer de quelle manière une MTA peut calculer H_{MT} et D_{MT} et montrer pourquoi cela ne conduit pas à une contradiction. Je montrerai ensuite contrairement à ce qu'affirment Svozil et Cotogno qu'une MTA ne peut pas calculer H_{MTA} et D_{MTA}.

2.2.2 Le calcul de H_{MT} et D_{MT} est-il contradictoire ?

Comme je l'ai expliqué précédemment, une MTA est identique à une MT sauf en ce qui concerne le nombre d'étapes qu'elle peut exécuter. Tandis que le nombre d'étapes d'une MT est toujours fini à la fin d'un calcul, la MTA peut quant à elle en exécuter une infinité en suivant la série géométrique suivante où i est la présente itération et n est le nombre d'itérations du calcul :

$$\sum_{i=0}^{n} \frac{1}{2^i}$$

Maintenant, voici la suite d'instructions qu'une MTA doit suivre pour calculer la fonction arrêt H_{MT} définie de la façon suivante : pour toute paire d'entiers x et y, $H_{MT}(x, y) = 1$ si x est le code d'une MT qui s'arrête lorsqu'on lui fournit la donnée en entrée y ; $H_{MT}(x, y) = 0$ autrement.

Commencer
Inscrire 0 dans la première cellule du ruban ;
$i := 1$;
Faire
Simuler la MT x avec la donnée y pendant i étapes ;
Si m s'arrête, inscrire 1 dans la première cellule du ruban ;
$i := i + 1$
Tant que $i > 0$
Fin

Que la MTA puisse calculer H_{MT} n'est pas du tout contradictoire pour deux raisons. La première raison est qu'une MTA, même si elle est définie

structurellement comme une MT, n'en est pas une ; elle ne remet donc pas en cause le résultat selon lequel H_{MT} n'est pas calculable par MT. La seconde raison est que la contradiction déduite dans la preuve de l'impossiblité pour une MT de calculer sa propre fonction arrêt repose sur le fait que le non-arrêt d'une MT est ambigue : on ne sait pas si cette dernière n'a pas encore trouvé de solution ou s'il n'existe pas de solution. La MTA surmonte quant à elle cette contradiction par sa capacité à toujours s'arrêter même si la MT qu'elle est en train de simuler ne s'arrête pas sur la donnée y.

Le fait qu'une MTA puisse calculer H_{MT} sans impliquer de contradiction n'est pas très étonnant mais on peut aller plus loin dans le raisonnement en considérant la fonction diagonale D_{MT} propre aux MT :

> La plupart des modèles d'hyper-calcul proposés peuvent calculer la fonction arrêt des machines de Turing. Même si le calcul de la fonction arrêt n'est pas essentiel pour le concept d'hyper-calcul (il suffit de calculer au moins une fonction non récursive), c'est une capacité plus que souhaitable. La plupart de ces hyper-machines peuvent néanmoins aller un peu plus loin et calculer $g(x)$ pour les machines de Turing $[D_{MT}(x)]$. Est-ce que cette puissance conduit à une contradiction logique ? [Kieu and Ord, 2005, p. 148].

Kieu et Ord répondent à cette question à l'aide de deux arguments. Le premier est identique à celui énoncé plus haut à propos de la fonction arrêt propre aux MT, à savoir que le comportement d'une MTA qui peut calculer D_{MT} est bien défini. Le comportement de l'HM revient en effet à s'arrêter sur l'entrée x - et à inscrire 0 - si et seulement si la x-ième MT ne s'arrête pas sur x. Ce comportement n'implique pas de contradiction car l'origine de l'impossiblité pour une MT de calculer D_{MT} - le calcul d'un MT à partir de son propre code - ne peut pas être invoquée dans le cas d'une MTA puisque cette dernière n'est pas une MT.

Le second argument est plus subtil et peut être reformulé comme ceci : le calcul de D_{MT} par une MTA n'est pas contradictoire car une MT peut elle aussi calculer une fonction diagonale propre à une classe de machines dont la puissance calculatoire lui est inférieure. Le calcul de la fonction diagonale propre à une classe de machines serait donc toujours possible par certaines

machines plus puissantes.

En effet, soient $\{\psi_i\}$ l'énumération de toutes les fonctions primitives récursives et G_{PR} la fonction diagonale propre aux fonctions primitives récursives définie par

$$G_{PR}(x) : \psi_x(x) + 1$$

Cette fonction n'est pas primitive récursive : si G_{PR} était primitive récursive, elle devrait être égale à ψ_k pour un certain k et dès lors, on devrait avoir $G_{PR}(k) = \psi_k(k)$. Or par définition, $G_{PR}(k) = \psi_k(k) + 1$. G_{PR} n'est donc pas primitive récursive. Cependant, ψ_x est parfaitement définie et certaines fonctions récursives sont capables de la calculer. Plus encore, de telles fonctions sont aussi capables de caculer G_{PR} [18]. Ainsi, que G_{PR} ne puisse pas être une fonction primitive résursive n'est pas contradictoire avec le fait qu'elle soit calculable par une fonction récursive - ou une MT.

Des constructions similaires montrent que H_{MT} et G_{MT} peuvent être calculées par des machines plus puissantes que la MT et donc que le calcul de telles fonctions n'est pas problématique. L'étude de ces machines a par exemple permis de fonder la théorie des *degrés de Turing* qui décrit le degré d'indécidabilité d'un ensemble donné à partir de la notion d'oracle [Rogers, 1987], [Odifreddi, 1989]. Par conséquent si un type de machines donné n'est pas capable de calculer sa propre fonction diagonale, cela ne signifie pas qu'il n'existe pas un autre type de machine qui puisse la calculer.

Rentrons maintenant au centre du problème posé par Cotogno et Svozil : si une MTA est capable de calculer ses propres fonctions arrêt ou diagonale elle devrait être considérée comme contradictoire. Commençons par la fontion arrêt propre aux MTA, à savoir H_{MTA}.

18. Il existe en effet une procédure effective pour calculer G_{PR} : il suffit de calculer $\psi_x(x)$ - ce qui est possible car $\psi_x(x)$ est primitive récursive - et d'ajouter 1. Malgré l'existence de cette procédure, G_{PR} n'est pas primitive récursive.

2.2.3 Une MTA peut-elle calculer H_{MTA} ?

Pour Svozil et Cotogno, une MTA est logiquement contradictoire car elle est capable de calculer sa propre fonction arrêt [Svozil, 1998], [Cotogno, 2003]. Voici leur raisonnement. Tout d'abord, puisqu'elles sont structurellement identiques aux MT, les MTA possèdent deux propriétés fondamentales : la première est que chaque MTA peut être associée à un entier naturel à l'aide d'un codage adéquat, la seconde est qu'il existe une MTA dite *universelle* qui est capable de simuler une MTA particulière travaillant sur une donnée en entrée choisie. Ensuite, puisque les MTA s'arrêtent toujours après un intervalle de temps fini, le problème de l'arrêt peut être trivialement résolu par une MTA universelle. Pour ce faire, la MTA universelle inscrit après un intervalle de temps fini le symbole 1 signifiant qu'une MTA particulière s'arrête après un intervalle de temps fini - ce qui est toujours le cas. Par conséquent une MTA est capable de calculer H_{MTA}.

Si le résultat issu du raisonnement de Svozil et Cotogno est correct, il serait la preuve que les MTA sont logiquement contradictoires. D'après Turing effet, une HM qui a la possibilité de calculer sa propre fonction arrêt est nécessairement contradictoire [Turing, 1939]. Un tel résultat remettrait donc en cause les HM pouvant exécuter un nombre infini d'étapes en un temps fini et serait plus généralement une menace pour la possiblité logique de l'hyper-calcul. Comment donc remédier à ce problème si remède il y a ?

Il existe une solution qui a été proposée par Copeland et Shagrir et qui consiste à défendre que l'argument de Svozil et Cotogno ne remet pas en cause le calcul des MTA [Copeland and Shagrir, 2011]. Les auteurs de cette solution soutiennent en effet qu'une MTA ne calcule pas H_{MTA} dans un sens conventionnel et donc qu'elle n'est pas capable de résoudre le problème de l'arrêt propre aux MTA.

De leur point de vue, une machine peut calculer une fonction de deux manières différentes : dans un *sens interne* ou dans un *sens externe*.

– Une fonction f est calculable par une machine dans un *sens interne* seulement si (1) elle peut produire $f(n)$ pour tout argument n appartenant à son domaine de définition et (2) si elle est capable d'indi-

quer qu'elle a produit le résultat soit en s'arrêtant après avoir fourni le résultat, soit par d'autres moyens - comme par exemple inscrire un symbole spécial. La méthode permettant d'indiquer le résultat est néanmoins soumise à une restriction : cette méthode ne doit pas faire apppel à un dispositif ou à un comportement qui est *externe* à la définition de la machine.

– Une fonction f est calculable par une machine dans un *sens externe* seulement si elle peut produire $f(n)$ pour tout argument n appartenant à son domaine de définition. La machine peut fournir le résultat en exécutant une action prédéfinie dans un intervalle de temps prédéfini - ouvert ou clos - qui est délimité par une entité de référence externe à la machine - c'est-à-dire qui n'est pas contenu dans sa définition.

En résumé, ce qui différencie le calcul interne du calcul externe est qu'une machine calculant dans un sens interne doit seulement produire $f(n)$ et inscrire le résultat. Une machine calculant dans un sens externe doit en revanche inscrire le résultat après un intervalle de temps prédéfini à l'avance par un utilisateur qui doit faire appel à un dispositif externe à la machine pour mesurer cet intervalle de temps - une montre par exemple.

Copeland et Shagrir ne remettent pas en cause le fait qu'une MTA soit capable de calculer H_{MTA} mais ils affirment qu'une MTA calcule H_{MTA} dans un sens externe. En effet, si l'on suppose qu'une étape de calcul met 1 seconde à être exécutée alors un utilisateur doit se munir d'un dispositif externe à la machine - un chronomètre par exemple - pour avoir accès après deux secondes au résultat d'une MTA qui calcule H_{MTA}.

Selon eux, le calcul de H_{MTA} dans un sens externe n'implique aucune contradiction logique. La raison en est que cette contradiction - établie par Turing dans le cas des o-machines - ne s'applique que si une machine calcule sa propre fonction arrêt dans un sens interne : une fonction est en effet Turing-calculable uniquement si une MT est capable de calculer dans un sens interne, c'est-à-dire sans référence externe à la machine. Ainsi ce n'est qu'en prouvant qu'une MTA est capable de calculer sa propre fonction arrêt dans un sens interne que l'on peut déduire qu'elle est logiquement contradictoire.

L'argument apporté par Copeland et Shagrir est de plus confirmé par

Potgieter qui a fourni une preuve que les MTA ne sont pas capables de calculer leur propre fonction arrêt dans un sens interne [Potgieter, 2006]. L'idée générale est de proposer une classe de machines qui ne peut pas résoudre le problème de l'arrêt propre à leur classe et de montrer que les MTA appartiennent à cette classe. J'expose cette preuve dans l'annexe D.

L'argument de Svozil et Cotogno ne remet donc pas en cause les MTA puisque ces dernières ne sont pas capables de calculer leur propre fonction arrêt dans un sens interne. Il reste cependant un second problème lié à la possibilité pour une MTA de calculer sa propre fonction diagonale.

2.2.4 Une MTA peut-elle calculer D_{MTA} ?

Considérons en premier lieu les fonctions arrêt et diagonale propres aux MTA définies d'une façon différente que précédemment.

Définition (Fonctions arrêt et diagonale propres aux MTA)
Soient $\psi_1, \psi_2, \psi_3 \ldots$ une énumération des fonctions calculables par MTA. Soient H_{MTA} la fonction arrêt propre aux MTA définie par

$$H_{MTA}(x,y) = \begin{cases} 0 \text{ si } \psi_x(y) \nearrow \\ 1 \text{ si } \psi_x(y) \searrow \end{cases}$$

et D_{MTA} la fonction diagonale propre aux MTA définie par

$$D_{MTA}(x) = \begin{cases} 0 \text{ si } \psi_x(x) \nearrow \\ \nearrow \text{ si } \psi_x(x) \searrow \end{cases}$$

Avec $\nearrow$ et $\searrow$ signifiant respectivement que la fonction diverge ou converge.

Ensuite, définissons D'_{MTA} en substituant $\neg\psi_x(x)$ à $\psi_x(x)$:

$$D'_{MTA}(x) = \begin{cases} 1 \text{ si } \neg\psi_x(x) \searrow \\ \nearrow \text{ si } \neg\psi_x(x) \nearrow \end{cases}$$

A partir de cette nouvelle fonction on peut conclure que si une MTA pouvait calculer à la fois $\psi_x(x)$ et $\neg\psi_x(x)$ alors elle pourrait calculer à partir de D_{MTA} et de D'_{MTA} sa propre fonction diagonale : elle inscrirait en effet 0

si $\psi_x(x) \nearrow$ et 1 si $\neg\psi_x(x) \searrow$. Autrement dit, si une MTA pouvait calculer à la fois $\psi_x(x)$ et $\neg\psi_x(x)$ alors elle serait contradictoire.

Est-ce qu'une MTA est capable de calculer à la fois $\psi_x(x)$ et $\neg\psi_x(x)$? En premier lieu, puisqu'une MTA est capable de calculer sa propre fonction arrêt, $\psi_x(x)$ est calculable par MTA. Mais qu'en est-il de $\neg\psi_x(x)$? A première vue, $\neg\psi_x(x)$ semble être elle aussi calculable car la négation est une opération calculable et $\psi_x(x)$ est une fonction calculable.

En fait, $\neg\psi_x(x)$ n'est pas calculable par une MTA car les fonctions calculables par MTA sont les fonctions Σ_1^0 - les fonctions récursivement énumérables *cf.* annexe A - qui ne sont pas closes par composition. Autrement dit, si f et g sont deux fonctions calculables par MTA il n'est pas toujours le cas que la composée de f et g soit elle aussi calculable. Par conséquent $\neg\psi_x(x)$ n'est pas calculable par MTA. Voici une preuve qui montre que les fonctions calculables par MTA sont les fonctions Σ_1^0.

Théorème

Si $\mathcal{P}(MTA)$ *est l'ensemble des prédicats sur* $\mathbb{N}$ *décidables par MTA, alors*

$$\mathcal{P}(MTA) = \Sigma_1^0$$

.

Preuve :

$\mathcal{P}(MTA) \supseteq \Sigma_1^0$ Tout prédicat dans Σ_1 peut être exprimé sous la forme d'une formule de type $\exists y R(x, y)$ où R est un prédicat récursif. Une MTA travaillant sur la donnée en entrée x peut calculer $R(x, 1)$ dans un intervalle de temps fini et fournir un résultat si $R(x, 1)$ est vrai. Si $R(x, 1)$ est faux, la MTA continue de fonctionner pendant un autre intervalle en calculant $R(x, 2)$, $R(x, 3)$ et ainsi de suite jusqu'à trouver une instance vraie. Enfin, après deux unités de temps - 2 secondes par exemple - la première cellule du ruban contiendra 1 si et seulement si $\exists y R(x, y)$. Par conséquent tout prédicat dans Σ_1 possède une MTA qui lui est équivalent.

$\mathcal{P}(MTA) \subseteq \Sigma_1^0$ Etant donnée une MTA, soit $R_{MTA}(x, y)$ un prédicat qui est vrai si et seulement si l'HM travaille sur x et inscrit le résultat dans la

première cellule du ruban après y intervalles de temps. $\exists y R_{MTA}(x, y)$ est vraie si et seulement si la MTA inscrit 1 lorsqu'elle travaille sur x ; et est fausse si elle inscrit 0. Puisque $R_{MTA}(x, y)$ est un prédicat récursif pour toute MTA, $\exists y R_{MTA}(x, y)$ est dans Σ_1 pour toute MTA. Ainsi chaque MTA possède un prédicat dans Σ_1 qui lui est équivalent □

En conclusion, les arguments proposés par Svozil et Cotogno ne parviennent pas à démontrer que les MTA sont logiquement contradictoires. Premièrement, même si ces dernières calculent leur propre fonction arrêt, ce calcul est réalisé dans un sens externe ce qui n'implique pas de contradiction. La raison en est que les MTA ne sont pas capables de calculer cette fonction dans un sens interne, sens sur lequel repose la démonstration de Turing montrant qu'une MT calculant sa propre fonction arrêt est contradictoire. Deuxièmement, ces HM sont aussi incapables de calculer leur propre fonction diagonale à cause de leur puissance calculatoire se limitant aux fonctions Σ_1^0. Les MTA échappent ainsi à une contradiction par diagonalisation.

Conclusion

Le premier chapitre de mon travail a eu pour but de montrer que l'hyper-calcul est une notion logique consistante. Dit autrement, j'ai tenté d'expliquer pourquoi la possibilité de calculer des fonctions non Turing-calculables n'est pas logiquement contradictoire.

La raison pour laquelle j'ai consacré le premier chapitre de ma thèse à la démonstration de la possibilité logique de l'hyper-calcul est qu'une telle démonstration se révèle être une étape nécessaire dans le but d'étudier sa possibilité physique. Comme je l'ai expliqué dans l'introduction de ce chapitre, la possibilité physique de l'hyper-calcul est dépendante de sa possibilité logique qui est la cible de deux sérieuses objections :

1. Une première objection qui est énoncée contre la notion même d'hyper-calcul. Cette objection affirme que l'hyper-calcul est logiquement contradictoire car elle est en opposition avec la thèse de Church-Turing (TCT).

2. Une seconde objection qui est quant à elle invoquée contre les processus infinis utilisés par les HM pour calculer des fonctions non Turing-calculables.

Pour montrer que l'hyper-calcul n'est pas logiquement contradictoire j'ai donc apporté une solution à ces deux objections. Voici en substance quelles sont ces solutions.

De mon point de vue, la première objection faite à l'hyper-calcul ne tient pas car l'hyper-calcul n'est pas en opposition avec la TCT. J'ai commencé par expliquer, à partir des travaux de Copeland, comment la suppression de certaines contraintes inscrites au sein de la définition de procédure effective permet de formaliser des machines - appelées *hyper-machines* (HM) - qui calculent plus de fonctions que la MT. J'ai ensuite reformuler le problème de l'opposition entre hyper-calcul et TCT en introduisant la notion de contrainte satisfaite par une machine : affirmer que l'hyper-calcul est en opposition avec la TCT revient à défendre la thèse selon laquelle certaines machines satisfaisant *toutes et seulement* les contraintes énoncées dans la définition de procédure effective peuvent calculer davantage de fonctions que la MT. Or les HM ne sont pas formalisées à partir des mêmes contraintes que la MT. L'hyper-calcul ne remet donc pas en cause la TCT.

De manière plus générale, ce résultat tend à considérer l'hyper-calcul et le calcul effectif comme des définitions alternatives de la notion de calcul. En ce sens, il n'existe pas *une* définition de la notion de calcul mais plusieurs définitions à partir desquelles on peut formaliser des machines théoriques possédant des puissances calculatoires différentes. De ce point de vue, plusieurs notions de calcul peuvent être définies en fonction des contraintes que l'on introduit dans leur définition. Ces contraintes permettent de caractériser de façon précise la puissance calculatoire des machines que l'on peut formaliser. Il n'est donc pas contradictoire que le calcul effectif et l'hyper-calcul soient logiquement possibles.

La seconde objection, qui comprend deux arguments contre les processus infinis exécutés par les HM, ne parvient pas non plus à prouver que l'hyper-calcul est une notion logique contradictoire. Dans un premier temps, les paradoxes de l'infini sur lesquels s'appuie le premier des deux arguments,

ne permettent pas de conclure qu'il est logiquement contradictoire d'exécuter une infinité d'étapes en un temps fini, processus appelé *supertask* (ST). J'ai en effet tenté d'expliquer à partir des travaux de Benacerraf que la conclusion des raisonnements de Zénon, de Thomson et de Ross selon laquelle les ST sont contradictoires, ne pouvait pas être considérée comme vraie du fait qu'elle n'était pas une conséquence logique de ces raisonnements.

Dans un second temps, j'ai proposé certaines solutions afin d'échapper au second argument énoncé par Svozil et Cotogno contre les ST. Ces derniers défendent en particulier que les MTA - HM pouvant exécuter un ST - sont logiquement contradictoires en vertu du fait qu'elles sont capables de calculer leurs propres fonctions arrêt et diagonale. Il me semble néanmoins que Svozil et Cotogno ne parviennent pas à démontrer que les MTA sont logiquement contradictoires. Premièrement, même si ces dernières calculent leur propre fonction arrêt, ce calcul est réalisé dans un sens externe ce qui n'implique pas de contradiction. La raison en est que les MTA ne sont pas capables de calculer cette fonction dans un sens interne, sens sur lequel repose la démonstration de Turing montrant qu'une MT ou une HM calculant sa propre fonction arrêt est contradictoire. Deuxièmement, ces HM sont aussi incapables de calculer leur propre fonction diagonale pour la raison suivante : tandis que la fonction diagonale propre aux MTA n'est pas une fonction Σ_1, la puissance calculatoire des MTA se limitent aux fonctions Σ_1 proprement dites.

La démonstration de la possibilité logique de l'hyper-calcul peut à présent servir d'assise à une évaluation de sa possibilité physique et plus précisément de la thèse selon laquelle il est possible de construire physiquement une HM.

Chapitre 2

Pourquoi est-il nécessaire de construire physiquement une hyper-machine ?

> Miss Ambrose says it is logically impossible [for a man] to run through the whole expansion of π. I should have said it was *medically* impossible.
>
> Hilary Putnam

.

J'ai défendu dans le chapitre précédent la thèse selon laquelle l'hyper-calcul est logiquement possible. Pour ce faire, j'ai établi la non contradiction de l'énoncé suivant : il est possible de calculer une fonction non calculable par machine de Turing (non Turing-calculable). Ma démonstration s'est faite en deux temps. J'ai tout d'abord expliqué comment certaines machines théoriques nommées () sont capables d'hyper-calculer. J'ai ensuite tenté de défaire les arguments qui sont énoncés contre les processus utilisés par les HM pour surpasser la machine de Turing (MT) - l'exécution d'une infinité d'étapes de calcul en un temps fini en est un exemple.

L'existence - au sens mathématique - d'HM capables d'hyper-calculer est-

elle une preuve de la possibilité de franchir la barrière de Turing ? Autrement dit, atteste-t-elle du fait que les limites en principe admises depuis les années 1930 sont fausses ?

Ce chapitre a pour vocation de répondre par la négative à ces deux questions. Plus précisément, mon but est double : je souhaite d'une part montrer que la formalisation d'une HM n'est pas suffisante pour hyper-calculer ; je veux d'autre part défendre qu'il est indispensable de construire physiquement une HM si l'on souhaite franchir la barrière de Turing. Pour reprendre les mots de Welch,

> [...] chaque fois qu'un modèle d'hyper-calcul est proposé avec enthousiasme sans avoir été jugé préalablement sur les bases de la réalisabilité physique, il n'est pas effectif dans un certain sens [...] *calculer au-delà de la barrière de Turing* n'est dans ce cas pas un triomphe mais un subterfuge [Welch, 2004, p. 746].

La nécessité de construire une HM pour hyper-calculer a des conséquences philosophiques importantes. En particulier, elle permet d'affirmer que la limite du calculable n'est pas indépendante de la construction des ordinateurs. L'étude de ces machines deviendrait par conséquent indispensable pour définir les limites du calcul contrairement à ce qui a été soutenu depuis la seconde moitié du XX^{e} siècle.

L'analyse des logiciens des années 1930 a en effet permis de montrer que la construction d'ordinateurs n'avait aucune influence sur les limites du calcul. D'après cette analyse, les calculs exécutés par un ordinateur - aussi puissant soit-il - peuvent tous être effectués par un calculateur humain disposant de ressources physiques inépuisables ; l'être humain n'a donc pas besoin des ordinateurs puisqu'il peut en principe calculer les mêmes fonctions. Les ordinateurs ne sont ainsi utiles qu'en pratique lorsque les ressources nécessaires au calcul sont limitées.

Cependant, la distinction en principe / en pratique sur laquelle se fonde l'analyse précédente n'est selon moi pas pertinente pour déterminer ce qui est calculable. La raison en est qu'un calculateur humain disposant de ressources physiques inépuisables *n'est pas capable* d'effectuer tous les calculs - et plus précisément les hyper-calculs - exécutés par une HM physiquement

construite. En d'autres termes, le calcul de fonctions non Turing-calculables ne peut se faire sans une HM physiquement construite car l'être humain n'est en principe pas capable d'hyper-calculer. Que se soient en pratique ou en principe, la prise en compte d'un dispositif externe au calculateur humain est par conséquent indispensable pour étudier les limites du calculable.

Mon argumentation se fera en deux temps. Dans un premier temps, j'expliquerai pourquoi l'analyse de Turing et Church permet de faire abstraction des ordinateurs lorsque l'on étudie les limites du calculable. Je montrerai que les ordinateurs ont un intérêt pratique évident au sens où ils permettent d'exécuter des calculs infaisables manuellement mais qu'ils sont néanmoins négligeables en principe. Dans un second temps, je défendrai la position selon laquelle il est impossible de faire abstraction de la construction d'une HM si l'on souhaite franchir la barrière de Turing. Mon argument principal consistera à affirmer qu'il n'est pas possible d'hyper-calculer sans l'aide d'une HM physiquement construite, et cela même en principe - lorsque les ressources physiques ne sont pas limitées.

1 L'ordinateur n'est pas nécessaire pour calculer

Dans cette section, mon objectif est de montrer pourquoi l'analyse de Turing et Church permet de faire abstraction des ordinateurs lorsque l'on étudie les limites du calculable. Pour ce faire, je me reposerai sur une distinction entre *faisabilité pratique* et *faisabilité en principe*. Je défendrai plus précisément la thèse selon laquelle les êtres humains ont besoin des ordinateurs pour avoir accès aux résultats de calculs infaisables en pratique ; mais que les ordinateurs sont en revanche contingents pour avoir accès aux résultats de calculs faisables en principe.

Ma démonstration suppose toutefois de répondre à la question posée par Searle en 1992 : « qu'est-ce qu'un ordinateur exactement ? » [Searle, 1992, p. 205]. Répondre à cette question est en effet essentiel car la relation entre l'ordinateur et la faisabilité peut différer en fonction de ce que l'on considère

comme étant un ordinateur. Cette idée peut être illustrée par un exemple très simple. Supposons que les ordinateurs ont pour fonction de repousser les capacités calculatoires de l'être humain en fournissant les résultats de certains calculs infaisables manuellement. Peut-on défendre cette hypothèse si l'on considère les abaques comme des ordinateurs ? La réponse est négative car d'après moi les abaques ne sont pas des ordinateurs au sens où ils ne sont pas utilisés pour fournir les résultats de calculs infaisables manuellement mais pour aider le calculateur dans son calcul - en lui faisant gagner du temps ou en limitant les erreurs de calcul.

1.1 Qu'est-ce qu'un ordinateur ?

Il n'existe pas en informatique de réponse unanime à la question de savoir ce qu'est un ordinateur. Une des réponses que l'on peut fournir au premier abord est qu'un ordinateur est un *type particulier de mécanisme qui calcule*. Cette réponse, si simple soit-elle, est déjà pleines de présupposés. En effet, considérons le passage suivant :

> Il n'y a pas de propriété intrinsèque qui soit nécessaire et suffisante à tout ordinateur [...] Il est concevable que les tamis et les batteuses soient considérés comme des ordinateurs pour autant que quelqu'un ait une raison de s'intéresser à la fonction spécifique qui est représentée par leur comportements entrée-sortie [Churchland and Sejnowski, 1992, pp. 65-66].

Si Churchland et Sejnowski ont raison, il devient alors faux de définir un ordinateur comme un type de mécanisme qui calcule est faux puisqu'il n'existe aucune distinction entre (1) les mécanismes qui calculent et ceux qui ne calculent pas et (2) les types de mécanismes qui calculent dont fait partie l'ordinateur. Il n'existe donc de ce point de vue aucune différence entre un ordinateur et une télévision ou entre un ordinateur et un boulier.

Le premier problème, à savoir celui de déterminer s'il existe une distinction entre les mécanismes qui calculent et ceux qui ne calculent pas, dépasse le cadre de cette thèse ; le lecteur peut néanmoins consulter [Piccinini, 2010] pour une présentation générale du problème. Le second problème qui est de

déterminer s'il existe plusieurs types de mécanismes de calcul est quant à lui pertinent pour mon sujet puisqu'une réponse à ce problème implique de définir précisément la notion d'ordinateur.

De mon point de vue, il existe trois types de mécanismes qui calculent :

1. Les *mécanismes d'aide au calcul* pour lesquels l'utilisateur est obligé de déplacer certains composants à la main à chaque utilisation. Exemples : tous les types d'abaques.
2. Les *machines à calculer* ou *calculateurs* dont la fonction est d'exécuter un ensemble restreint d'opérations sur des données en entrée de tailles prédéfinies. Exemples : la pascaline, la machine à différence de Babbage, les calculatrices de poche.
3. Les *ordinateurs* qui peuvent possséder plusieurs propriétés fondamentales telles que la programmabilité ou l'universalité. Exemples : la machine analytique de Babbage, l'ENIAC, les ordinateurs modernes.

Etudier en détails les mécanismes d'aide au calcul n'est pas très intéressant car en général personne n'affirmera qu'un boulier est un ordinateur. En revanche, la distinction entre calculateurs et ordinateurs est plus subtile et nécessaire pour mieux cerner le rôle de la construction des ordinateurs.

1.1.1 Les calculateurs

Un calculateur est un mécanisme de calcul qui manipule des *suites de symboles* et qui possède quatre composants : des *dispositifs pour les données en entrée* (D_e), des *dispositifs pour les données en sortie* (D_s), des *unités de mémoire* et des *unités de calcul*.

Les D_e du calculateur reçoivent deux types de données en entrée : les *données* et les *instructions de commande* qui commandent aux unités de calcul d'effectuer une certaine opération sur les données. Les instructions de commande sont généralement adressées au calculateur lorsque l'utilisateur appuie sur les boutons appropriés. Une *opération* est une transformation d'une donnée en résultats. Les unités de mémoire contiennent à la fois les données et les résultats intermédiaires jusqu'à ce que l'opération soit exécutée. Les unités de calcul exécutent quant à elles une opération sur les

données. L'opération qui est exécutée dépend uniquement de l'instruction de commande qui a été adressée aux D_e. Après que l'opération ait été exécutée, les D_s fournissent les résultats à l'utilisateur qui sont les valeurs de fonctions Turing-calculables.

La différence majeure entre calculateurs et ordinateurs se situe au niveau de leur puissance de calcul respective. Contrairement aux ordinateurs, la puissance de calcul des calculateurs est limitée par trois caractéristiques importantes :

1. Les calculateurs ne sont pas *programmables* car on ne peut pas les modifier pour qu'ils suivent des instructions différentes de celles qui leurs ont été attribuées lors de leur construction. Précisément, puisque le nombre de ces instructions ne peut pas être augmenté en ajoutant de nouvelles instructions, les résultats fournis par un calculateur sont les valeurs d'un nombre fini de fonctions mathématiques qui ne peut pas être modifié. En effet, l'ensemble des fonctions calculables par un calculateur est déterminé par les opérations primitives pouvant être exécutées par les unités de calcul. La fonction calculée à un moment donné est donc uniquement déterminée par l'instruction de commande adressée au calculateur *via* les D_e. Le nombre de ces instructions de commande n'étant pas modifiables, le nombre de fonctions calculables par un calculateur est lui aussi non modifiable.
2. Les calculateurs ne sont pas *universels* : ils ne peuvent pas calculer l'ensemble des fonctions Turing-calculables. Lors d'un calcul, les calculateurs exécutent une seule opération sur leurs données, fournissent en sortie le résultat correspondant et s'arrêtent. Ils n'ont ni la capacité d'exécuter dans un certain ordre plusieurs de leurs opérations primitives, ni celle d'exécuter des instructions définies à partir des opérations primitives. Ces opérations peuvent néanmoins être assemblées en une suite d'instructions mais uniquement *via* l'insertion d'instructions de commande après l'exécution de chaque opération. En d'autres termes, les calculateurs n'ont pas de structure de contrôle et ne peuvent pas utiliser d'instructions telles que la condition (*si... alors*) ou la boucle (*tant que... faire*) qui sont nécessaires pour calculer l'ensemble des fonctions

Turing-calculables.

3. Les calculateurs n'ont pas de *mémoire virtuelle* [Piccinini, 2008a]. Plus précisément, la taille de leur espace mémoire est préalablement fixée et l'ensemble des valeurs qu'ils peuvent calculer est limité par cet espace mémoire qui ne peut pas être augmenté. Ainsi, leurs D_e et D_f ne peuvent accepter et fournir que des suites de symboles ayant une taille maximale non modifiable.

En résumé, les calculateurs ne possèdent pas trois des principales propriétés que possèdent les ordinateurs actuels - programmabilité, universalité et flexibilité de la mémoire. Ces propriétés vont maintenant être explicitées en étudiant de plus près les ordinateurs.

1.1.2 Les ordinateurs

Comme les calculateurs, les ordinateurs sont constitués de quatre types de composants : des D_e, des D_s, des unités de mémoire et des unités de calcul appelés *processeurs*. La différence entre les calculateurs et les ordinateurs réside dans leurs propriétés et leur architecture interne. Les processeurs des ordinateurs sont par exemple capables d'exécuter toutes sortes d'opérations primitives dans un ordre arbitraire à l'aide de structures de contrôle telles que la condition ou la boucle. Leurs unités de mémoire peuvent quant à elles être augmentées si davantage d'espace mémoire est nécessaire [1].

Je présente ci-dessous les principales propriétés que peuvent posséder les ordinateurs. Ces propriétés ne sont toutefois ni nécessaires ni suffisantes pour caractériser parfaitement la notion d'ordinateur car un nombre important de machines construites depuis le début du 20^{e} siècle ont été appelées ordinateurs même si elles ne partagent pas toutes les propriétés des ordinateurs actuels - appelés généralement *ordinateurs modernes*. Le but de mon analyse est donc de présenter les principales caractéristiques que peuvent posséder les ordinateurs afin de fournir à la fin de cette analyse une définition correcte de ce qu'est un ordinateur moderne.

1. Cette mémoire est bien sûr limitée en dernier lieu par les lois de la physique [Lloyd, 2000]. Toutefois, la mémoire d'un ordinateur peut être modifiée à volonté tant qu'une telle limite physique n'est pas atteinte.

Programmabilité. Une machine est programmable lorsqu'elle peut être facilement modifiée pour se comporter de différentes façons. A partir de cette définition, on peut remarquer que des mécanismes tels les métiers à tisser ou certaines boîtes à musique peuvent être programmables même s'ils n'ont pas pour fonction de calculer. Un ordinateur est quant à lui programmable si l'on peut modifier la suite de ses instructions pour le faire exécuter de nouvelles instructions qui dépendent de la modification.

Ce qui peut être considéré comme une modification légitime dépend du contexte dans lequel on utilise la machine et des moyens disponibles pour faire la modification. Par exemple, on pourrait en théorie modifier certains calculateurs pour qu'ils calculent de nouvelles fonctions soit en les reconfigurant soit en leur rajoutant de nouveaux composants. Toutefois, ce type de modification ne saurait être considéré comme légitime car elles impliqueraient la construction de nouvelles machines plutôt que leur programmation.

Pour être plus précis, il existe deux formes de programmabilité qui dépendent du type de modification que l'on doit faire sur la machine afin de modifier son comportement :

1. La *programmation matérielle* implique le changement par le ou les utilisateurs de l'organisation spatiale des composants de l'ordinateur afin de modifier le nombre d'opérations ou l'ordre des opérations que peut effectuer la machine. Ces changements ont pour conséquence de modifier les valeurs qui seront fournies en sortie par l'ordinateur. Par exemple, certains des premiers ordinateurs tels que le célèbre ENIAC [2] étaient fabriqués à partir d'interrupteurs, d'écrans et de câbles qui envoyaient des signaux aux composants ayant pour fonction de calculer. Permuter les interrupteurs ou modifier la configuration des câbles avaient pour effet de connecter les différents composants de la machine de différentes manières pour que soit modifié son comportement calculatoire. De nos jours, la programmation matérielle qui est longue et fastidieuse est

2. L'ENIAC (*Electronic Numerical Integrator Computer*) est le premier ordinateur universel - terme qui est expliqué plus loin. Il a été conçu par Mauchly et Eckert de l'université de Pennsylvanie et fabriqué grâce aux financements de l'armée des Etats-Unis durant la seconde guerre mondiale. Pour plus d'informations sur l'ENIAC, consulter [Van der Spiegel et al., 2000].

remplacée par la programmation logicielle.

2. La *programmation logicielle* permet de modifier le comportement calculatoire des ordinateurs modernes à partir de suites de symboles traitées par certains de ses composants ayant pour fonction de changer les opérations qui sont exécutées par la machine. Ces suites de symboles peuvent être considérées comme des ensembles d'instructions appelés *programmes* que doit suivre l'ordinateur. La programmation logicielle, contrairement à la programmation matérielle, ne peut donc se faire qu'à partir d'un réarrangement approprié des suites de symboles - instructions - sans apporter de modification manuelle à l'un de ses composants. En particulier, on peut distinguer deux types de programmation logicielle :

   (a) *La programmation logicielle externe* requiert que le programme soit inséré au sein de la machine par l'intermédiaire d'un D_e car cette dernière ne possède pas la capacité de copier et de mémoriser le programme en question. Par exemple, l'Harvard Mark I [3] ne pouvait être programmé que de façon externe.

   (b) *La programmation logicielle interne* requiert que la machine possède des composants ayant pour fonctions de copier et de mémoriser les programmes. Cette forme de programmation est à la fois plus flexible et plus efficace que la précédente et elle est devenue la forme standard de programmation des ordinateurs modernes.

Universalité. Certains anciens ordinateurs tels que l'ABC [4] qui n'ont pas de programmation interne ont été conçus dans le but de réaliser des tâches spécifiques, c'est-à-dire pour ne calculer que certaines fonctions parmi l'ensemble des fonctions Turing-calculables. Ces ordinateurs - qui ne peuvent

3. L'Harvard Mark I a été conçu par Aiken en 1944 et pouvait être programmé à l'aide de relais et de bandes perforées pour calculer certaines fonctions mathématiques générales. Pour plus de détails, voir [Cohen, 1999].

4. L'ABC (*Atanasoff-Berry Computer*) a été créé en 1942 pour résoudre des systèmes de plus de 29 équations algébriques linéaires à 29 inconnues [Atanasoff, 1940]. D'après Burks, ce serait la première machine à avoir été appelée *ordinateur* [Burks and Burks, 1988].

suivre qu'un nombre restreint d'algorithmes - sont encore fréquemment utilisés pour certaines applications ; les ordinateurs que l'on trouve dans les automobiles en sont des exemples.

En outre, plusieurs ordinateurs pouvant être programmés de manière plus flexible ont été fabriqués depuis les années 1940 [Rojas and Hashagen, 2000]. Ces ordinateurs ont la capacité de calculer de plus grande ensembles de fonctions et sont généralement appelés *ordinateurs universels* [von Neumann, 1945]. Déterminer si un ordinateur possède la propriété d'universalité dépend généralement de sa quantité d'espace mémoire et de l'aisance avec laquelle on peut le programmer. Plus particulièrement, les ordinateurs universels ont la capacité de modifier leurs programmes afin de calculer l'ensemble des fonctions calculables par MT. Bien entendu, ces ordinateurs calculent toutes les fonctions Turing-calculables uniquement dans un sens *théorique* puisque ces machines devront s'arrêter si un calcul demande trop de ressources physiques pour être exécuté jusqu'au bout. Les ordinateurs de bureau, les ordinateurs portables ou même la machine analytique de Babbage [5] sont des exemples d'ordinateurs universels.

Hiérarchie fonctionnelle. Pour qu'un ordinateur soit universel, c'est-à-dire pour qu'il puisse exécuter n'importe quel programme opérant à l'aide d'une notation donnée, il est nécessaire de coder la notation et le programme en utilisant les instructions du langage de la machine.

Dans le cas des premiers ordinateurs, les programmeurs devaient procéder à un tel codage manuellement. Ce codage était lent, fastidieux et sujet à de nombreuses erreurs. Pour augmenter la vitesse d'exécution du processus et diminuer le nombre d'erreurs, les programmeurs eurent l'idée de mécaniser certaines parties du processus de codage, ce qui a donnée lieu à la création d'une hiérarchie fonctionnelle au sein de l'organisation de l'ordinateur. Plus précisément, la mécanisation du codage peut se faire à partir de programmes exécutés par les processeurs de la machine tels que les *assembleurs* et les

5. La machine analytique a été proposée par Babbage en 1834 mais sa construction n'a jamais été complètement achévée. Cette machine est tout de même universelle si l'on opère quelques modifications mineures [Morrison and Morrison, 1961].

compilateurs. Ces programmes ont en particulier capables de produire de la *mémoire virtuelle*, des *notations complexes* et des *opérations complexes* qui permettent de programmer de manière plus rapide et plus fiable. Voici une description de ces trois importantes notions.

Lorsqu'un processeur exécute des instructions, des espaces mémoires appelés *registres* contenant les données et les instructions sont identifiés par la machine à partir *d'adresses*. Chaque registre possède une capacité de stockage fixée qui dépend du nombre de cellules qu'il contient - un registre peut être visualisé comme le ruban d'une MT qui serait divisé en un nombre fini de cellules. La *mémoire virtuelle* est une manière d'identifier les données et les instructions à l'aide d'adresses virtuelles qui sont indépendantes de leurs localisations physiques. Durant le calcul d'un ordinateur, les adresses physiques des registres en question sont automatiquement générées par le compilateur en fonction des adresses virtuelles. Enfin, même si la mémoire des registres n'a pas de limite théorique du fait que la mémoire virtuelle est identifiée indépendamment de sa localisation physique, le nombre de symboles que peut en pratique mémoriser un ordinateur dépend de la taille de sa mémoire physique.

Pour qu'un processeur exécute des instructions, toutes les données et les instructions doivent être décrites sous la forme de suites binaires - de 0 et de 1 - correspondant aux signaux physiques qui se déplacent à travers le processeur. Ces suites de données et d'instructions ont une longueur fixée qui correspond à la taille du registre qui les contient. Les *notations complexes* sont en outre des suites de symboles pouvant contenir n'importe quel élément d'un alphabet fini tel que l'alphabet latin. Ainsi, les programmeurs peuvent utiliser ces notations complexes pour former des structures de données pouvant être facilement interprétées et créer des langages similaires à leur langue naturelle. Lors d'un calcul, le codage des structures de données écrites sous la forme de notations complexes en données écrites sous la forme de 0 et de 1 - langage machine - est exécuté automatiquement par le compilateur et l'assembleur de la machine.

Même si ces suites peuvent être assemblées pour former des suites de tailles arbitraires grâce à la mémoire virtuelle, le processeur ne peut néanmoins

recevoir qu'un nombre fini d'instructions correspondant aux opérations primitives qu'il peut exécuter. Pour qu'un ordinateur soit universel, il faut par conséquent introduire des *opérations complexes* définies à partir d'opérations primitives ou à partir d'opérations complexes préalablement définies. Ces opérations complexes peuvent être exprimées en utilisant une notation complexe appelée *langage de programmation de haut niveau*[6] contenant des structures de contrôle telles que *si... alors* ou *tant que... faire* qui sont plus faciles à utiliser que le langage machine. Comme pour les autres notations complexes, les intructions écrites *via* le langage de haut niveau sont automatiquement codées en langage machine par le compilateur et l'assembleur.

Ordinateurs digitaux et analogiques. Les ordinateurs modernes sont des *ordinateurs digitaux* au sens où ils manipulent uniquement des suites finies de symboles. Toutefois, d'autres ordinateurs appelés *ordinateurs analogiques* ont été développés avant l'avènement des ordinateurs modernes. Définir précisément la notion d'ordinateur analogique est une entreprise qui dépasse largement le cadre de cette thèse ; c'est pourquoi je préfère présenter certaines distinctions conceptuelles qui me semblent essentielles pour expliquer ce qui différencie les ordinateurs analogiques des ordinateurs digitaux[7].

Tout d'abord, du point de vue de leur organisation, les ordinateurs digitaux et analogiques ne sont pas différents. Ces derniers sont aussi composés de D_e, de D_s et d'unités de calcul - et dans certains cas d'unités de mémoire. De même, la fonction des ordinateurs analogiques et identique à celle des ordinateurs digitaux, à savoir fournir des données en sortie à partir d'une procédure dont l'exécution dépend de la donnée en entrée. Il existe toutefois plusieurs différences entre ces deux types d'ordinateurs du point de vue de leur mécanique interne.

Tout d'abord, la distinction entre *système continu* et *système discret* n'est pas suffisante pour caractériser la différence entre ces deux types d'ordina-

6. Le premier langage de programmation de haut niveau a été créé par Zuse et se nomme *Plankalkül* [Wolfgang, 1997]. De nos jours, les ordinateurs modernes utilisent des langages de haut niveau tels que JAVA, PYTHON, et C.

7. La présentation repose sur trois ouvrages de références dans le domaine du calcul analogique : [Jackson, 1960], [Johnson, 1963], [Wilkins, 1970].

teurs. Les systèmes analogiques sont en effet souvent considérés comme étant continus et les systèmes digitaux comme étant discrets :

> La donnée en entrée d'un neurone est analogique (valeurs continues entre 0 et 1) [Churchland and Sejnowski, 1992, p. 51].

Cependant, fonder la différence entre systèmes analogiques et systèmes digitaux sur l'opposition continu /discret n'est pas suffisant car un système est considéré comme discret ou continu *relativement* à une certaine description mathématique du système qui s'applique à un certain niveau d'analyse. Par exemple, des auteurs tels que Toffoli [Toffoli, 1984] et Wolfram [Wolfram, 2002] défendent la thèse selon laquelle notre univers physique est discret à son niveau de description le plus fondamental - les ordinateurs analogiques ne pourraient ainsi pas exister à ce niveau fondamental. Par ailleurs, puisque certains domaines de la physique appliqués à l'ingénieurie tels que la physique des matériaux étudient des objets au niveau macroscopique à l'aide d'équations différentielles qui supposent le caractère continu de ces objets, les ordinateurs digitaux ne pourraient donc pas exister à un niveau macroscopique. Pourtant, les notions d'ordinateurs analogiques et d'ordinateurs digitaux ont un sens bien précis en informatique ou en ingénieurie qui ne dépend pas des théories physiques. Par conséquent la dichotomie continu /discret semble uniquement satisfaisante relativement à un certain niveau de description des systèmes en question.

Ls ordinateurs digitaux et analogiques se différencient en revanche par leurs données en entrée et en sortie. Tandis que les données des ordinateurs digitaux sont des suites finies de symboles, les données des ordinateurs analogiques sont quant à elles des *variables réelles* [Pour-El, 1974]. Schématiquement, les variables réelles sont des grandeurs physiques qui (1) varient en fonction du temps (2) prennent des valeurs sur un intervalle continu mais borné et (3) varient de manière continue en fonction du temps. Par exemple, la tension du courant électrique au sein d'un câble electrique est une variable réelle.

La seconde différence réside dans les opérations exécutées sur ces données. Tandis que les ordinateurs digitaux exécutent des opérations définies sur des suites finies de symboles, les opérations des ordinateurs analogiques sont

définies sur des variables réelles. Plus précisément, les ordinateurs analogiques et les unités de calcul ont pour fonction de transformer une variable réelle prise comme donnée en entrée en une autre variable réelle qui sera la donnée en sortie. Toutefois, cette transformation ne peut pas être exécutée à partir d'opérations discrètes - où chaque étape de la transformation est effectuée à chaque unité de temps - mais doit se faire de manière continue ce qui implique une modification continue de la variable réelle en fonction du temps. Enfin, tandis que cette transformation s'opère à partir d'une suite d'instructions sur les suites de symboles dans le cas des ordinateurs digitaux, la transformation d'une variable réelle s'effectue au sein des ordinateurs analogiques *via* un système d'équations différentielles.

Que les données manipulées par les ordinateurs analogiques soient des variables réelles conduit à des limitations physiques qui n'affectent pas les ordinateurs digitaux. Les ordinateurs analogiques sont en particulier moins précis que les ordinateurs digitaux. D'une part, deux variables réelles ne peuvent être distinguées - par la machine ou par un observateur externe - que conformément à un certain degré de précision lié à la préparation du processus de mesure. Au contraire, deux variables discrètes - par définition finies - peuvent toujours être distinguées de façon explicite - il suffit de comparer un à un les symboles de la suite.

Les ordinateurs analogiques peuvent enfin être divisés entre ordinateurs non universels et ordinateurs universels. La différence entre ces deux types d'ordinateurs analogiques réside dans la classe des équations différentielles qu'ils peuvent résoudre : les ordinateurs universels peuvent résoudre une plus grande classe d'équations différentielles que les ordinateurs non universels. Toutefois, la propriété d'universalité n'est pas exactement la même que celle que l'on attribue aux ordinateurs digitaux.

En effet, un ordinateur digital universel est capable de calculer l'ensemble des fonctions Turing-calculables qui sont par définition des fonctions de $\mathbb{N}$ dans $\mathbb{N}$. Plus précisément, un tel ordinateur est capable de coder par un symbole chaque argument d'une fonction Turing-calculable puis de produire - à partir de son programme - un autre symbole qui correspond à la valeur de la fonction appliquée à chaque argument. Cependant, les ordinateurs ana-

logiques ne manipulent pas de suites de symboles mais des variables réelles, il est donc pertinent de se demander en quel sens on peut dire qu'ils sont universels.

Puisque les ordinateurs analogiques ne travaillent pas sur des suites de symboles, on ne peut pas appliquer directement la notion de fonction Turing-calculable dans le but d'affirmer qu'ils sont universels. Il est néanmoins possible de remédier à ce problème en employant la notion de *fonction sur les réels*. En améliorant les travaux de Shannon [Shannon, 1941], Pour-El a identifié précisément quelles fonctions sur les réels peuvent être calculées par un ordinateur analogique qui est universel [Pour-El, 1974]. Ces fonctions sont les *fonctions algébriques différentielles*, c'est-à-dire les fonctions qui sont des solutions pour les *équations algébriques différentielles*[8]. Une équation algébrique différentielle est dite *universelle* si toute fonction continue définie sur les réels peut être approximée avec une précision arbitraire sur l'axe du temps $0 \leq t \leq \infty$ par une solution de l'équation. A partir de cette définition, des ordinateurs analogiques universels ont été conçus avec quatre *intégrateurs*[9] dont les données en sortie peuvent approximer n'importe quelle valeur d'une fonction continue sur les réels [Duffin, 1981].

En résumé, les ordinateurs analogiques ne peuvent pas être considérés comme des ordinateurs modernes pour deux raisons principales. La première est que les ordinateurs analogiques manipulent des données sous la forme de variables réelles et non sous la forme de suites finies de symboles. La seconde est qu'ils manipulent ces données à partir de transformations continues - par exemple des intégrations - contrairement aux ordinateurs modernes qui effectuent des opérations discrètes sur ces données. Il est à présent temps de proposer une réponse à la question posée en introduction : qu'est-ce qu'un ordinateur moderne ?

8. Ces équations sont de la forme $P(y, y', y'', \ldots, y^{(n)}) = 0$, où P est un polynôme dont les cœfficients sont des entiers, et où y est une fonction en x.

9. La technique la plus efficace pour résoudre des équations différentielles consiste à exécuter des intégrations successives sur les variables réelles. Les intégrateurs sont des composants qui ont ainsi pour fonction de produire en sortie une variable réelle qui est l'intégrale de la donnée fournie en entrée.

Conclusion : qu'est-ce qu'un ordinateur moderne ? Après avoir présenté les différentes propriétés que pouvaient posséder les ordinateurs, je propose une définition du concept d'ordinateur moderne. Un ordinateur moderne est un ordinateur qui possède les caractéristiques suivantes :

- Il est digital, c'est-à-dire qu'il manipule uniquement des suites de symboles qui sont transformées lors d'un calcul *via* une suite d'instructions.
- Ses données et ses instructions - ou programmes - sont décrites sous la forme de notations complexes à partir d'opérations complexes et sont mémorisées grâce à une mémoire virtuelle.
- Son comportement calculatoire peut être modifié à l'aide de langages de programmation de haut niveau mémorisés dans certains de ses composants. Il répond ainsi à une programmation logicielle interne.
- Il possède une mémoire virtuelle qui signifie (1) que sa mémoire physique n'est pas limitée en principe et (2) que la taille de sa mémoire physique peut être modifiée.
- La mémoire virtuelle et l'utilisation de langages de programmation de haut niveau ont pour conséquence de le rendre universel : l'ensemble des fonctions qu'il peut calculer est l'ensemble des fonctions calculables par MT.

La première étape qui vient d'être complétée - étape qui a consisté à définir de manière précise les ordinateurs actuels - permet à présent d'analyser la relation entre ces ordinateurs et l'accès aux résultats effectivement calculables.

1.2 Pourquoi construit-on des ordinateurs ?

Jusqu'aux années 1940, les calculs ne sont pas exécutés par des ordinateurs mais par des êtres humains dont le travail consiste à calculer [Copeland, 2000b]. Ces calculateurs humains calculent à l'aide de crayons et de feuilles et sont employés par différents types d'établissements tels que des entreprises commerciales ou le gouvernement. Ce n'est qu'en 1942 qu'une machine, à savoir l'ABC (*Atanasoff-Berry Computer*), est appelée *ordinateur* [Burks, 2002].

Une des raisons pour lesquelles cette machine est appelée *ordinateur* est que l'ABC peut exécuter, contrairement aux calculateurs ou autres abaques, des calculs qui dépassent les capacités calculatoires de l'être humain. Les calculateurs et les aides au calcul - telles que les abaques - sont en effet utilisés pour *faciliter* les calculs, c'est-à-dire pour gagner du temps ou pour limiter les erreurs de calcul. Ils ne sont donc pas insdispensables pour calculer car un être humain peut effectuer les mêmes calculs à un facteur de temps près. L'ABC peut en revanche résoudre des systèmes de 29 équations à 29 inconnues, capacité qui fait de lui un outil indispensable pour la résolution d'équations.

Si l'on devait placer la frontière à partir de laquelle la construction d'ordinateurs est nécessaire, elle se situerait donc lorsque la machine n'est plus considérée comme une aide au calcul mais comme un outil nécessaire au calcul, c'est-à-dire au moment où les capacités du calculateur humain - vitesse, espace mémoire - sont insuffisantes pour exécuter les calculs. Cette frontière est néanmoins difficile à tracer car les chiffres illustrant les capacités calculatoires du cerveau humain sont loin d'être fiables. A titre d'exemple, les chiffres correspondant au nombre d'instructions exécutés par seconde par le cerveau humain sont compris entre 10^{13} et 10^{19} [Delahaye, 2003].

Pour tenter de définir plus précisément cette frontière, je procède à une distinction entre les calculs pouvant être effectués *en pratique* par l'être humain et les calculs pouvant être effectués *en principe* par l'être humain ; autrement dit entre la et [Franchette, 2011] :

Définition (Faisabilité pratique et faisabilité en principe)
La faisabilité pratique renvoie aux calculs pouvant être effectués manuellement par l'être humain ***en prenant en compte*** *les ressources physiques utilisées pour mener à bien son calcul.*
La faisabilité en principe renvoie aux calculs pouvant être effectués manuellement par l'être humain ***indépendamment*** *des ressources physiques utilisées pour mener à bien son calcul.*

Définition (Calcul pouvant être effectué manuellement)
Un calcul peut être effectué manuellement si un être humain peut produire le résultat de ce calcul sans l'aide d'aucune machine.

La différence entre ces deux types de faisabilité réside dans la prise en compte ou non des ressources physiques utilisées lors des calculs. Une *ressource physique* peut revêtir plusieurs formes : du temps pour calculer, des feuilles de papier sur quoi écrire, des crayons pour écrire ou même une certaine résistance physique à la fatigue. Lorsqu'un calcul est faisable en principe, cela signifie intuitivement que le calculateur a effectué ce calcul sans jamais manquer de feuilles, de crayons ou de temps. Dans le cas contraire, c'est-à-dire dans le cas où le calcul est faisable en pratique, le calculateur a effectué le calcul en disposant d'une quantité limitée de temps, de feuilles, *etc.*

La frontière à partir de laquelle la construction d'ordinateurs est nécessaire peut d'après moi être spécifiée à partir de ces deux types de faisabilité. Plus particulièrement, je soutiens que les ordinateurs ne sont pas nécessaires dans le cadre de la faisabilité en principe mais qu'ils sont en revanche indispensables pour dépasser les limites de la faisabilité pratique.

1.2.1 Les ordinateurs sont nécessaires pour dépasser les limites de la faisabilité pratique

De mon point de vue, les ordinateurs sont construits pour exécuter calculs ne pouvant pas être exécutés manuellement à cause d'un manque de ressources physiques. Autrement dit, ils sont construits pour dépasser les limites de la faisabilité pratique.

Ce point de vue doit néanmoins être explicité en partie à cause de l'imprécision du prédicat *ne pouvant pas être exécuté manuellement à cause d'un manque de ressources physiques*. Ce prédicat est en effet imprécis car il suppose (1) que l'on puisse quantifier les ressources physiques nécessaires à un calcul ; et (2) que l'on puisse définir à partir de quelle quantité de ressources un calcul ne peut plus être exécuté manuellement.

L'évaluation de la quantité de ressources nécessaire à l'exécution d'un calcul peut être faite à partir de la théorie de la complexité qui détermine les problèmes mathématiques pouvant être résolus avec des ressources de calcul limitées [Papadimitriou, 1994].

Les ressources les plus couramment utilisées dans les calculs séquentiels

- les calculs où les étapes sont exécutées les unes après les autres - sont le temps et l'espace [Cook, 1983]. Mais comment mesurer ces ressources ? La théorie de la complexité mesure par exemple la quantité de temps à partir du nombre d'étapes qu'effectue une machine théorique - telle que la MT - pour exécuter un algorithme sur une donnée en entrée. L'utilisation d'une machine théorique - par opposition à l'utilisation d'une machine physiquement construite - est justifiée par le fait que la conception de la théorie de la complexité dans les années 1960 a pour origine une volonté de comparer les algorithmes entre eux indépendamment de la vitesse de calcul des machines qui les exécutaient. L'efficacité des algorithmes était en effet dépendante du processeur de la machine, de son langage de programmation ainsi que du compilateur utilisé ; une approche indépendante des facteurs matériels était donc nécessaire pour évaluer leur efficacité.

Plus précisément, cette approche se fait en attribuant une *complexité* aux algorithmes en fonction de la taille de la donnée en entrée. En particulier, trois types de complexité peuvent être mesurés :

- **Une complexité au pire des cas** : complexité maximale dans le cas le plus défavorable.
- **Une complexité moyenne** : complexité à partir d'une répartition probabiliste des tailles des données [Levin, 1986].
- **Une complexité au meilleur des cas** : complexité minimale dans le cas le plus favorable.

Dans la pratique, c'est la complexité au pire des cas qui est la plus utilisée car on souhaite généralement borner le temps d'exécution d'un calcul. La complexité en temps peut par exemple être formalisée à partir de la MT en mesurant la quantité de temps à partir du nombre de transitions - notée $f(n)$ - que la machine effectue lors d'un calcul. Il est néanmoins très difficile de calculer $f(n)$ de façon exacte et on se contente le plus souvent d'une *analyse asymptotique* en utilisant la notation O :

Définition (Notation O)
$f(n) = O(g(n))$ *si et seulement si* $\exists c, n_0 : \forall n \leq n_0 \quad f(n) \leq c.g(n)$

Etudions en guise d'exemple la complexité au pire des cas de l'algorithme

suivant qui permet à une MT de déterminer si la donnée en entrée inscrite sur son ruban fait partie de l'ensemble $A = \{0^k 1^k / k \geq 0\}$. Pour ce faire, la MT suit les instructions suivantes et inscrit Y si l'entrée fait partie de l'ensemble et N autrement :

1. La MT parcourt le ruban et inscrit N si un 0 est trouvé à droite d'un 1.
2. Tant qu'il reste au moins un 0 et un 1, la MT parcourt son ruban et efface à chaque passage un 0 et un 1.
3. S'il ne reste plus qu'un seul 1 ou un seul 0 alors elle inscrit N ; si son ruban est vide alors elle inscrit Y.

Calculons maintenant la complexité de l'algorithme où n est la taille de la donnée en entrée :

- Etape 1 : $2n = O(n)$
- Etape 2 : $(n/2) \times O(n) = O(n^2)$
- Etape 3 : $O(n)$
- Total : $O(n) + O(n^2) + O(n) = O(n^2)$

La complexité que l'on vient d'attribuer peut toutefois dépendre fortement du modèle théorique que l'on considère. C'est ainsi que la reconnaissance des palindromes - des mots où l'ordre des lettres ne change pas quel que soit le sens de lecture - peut se faire en temps linéaire par une MT possédant 2 rubans mais exige $O(n^2)$ étapes de calcul sur une MT à un seul ruban. Les principales machines que l'on considère pour mesurer la complexité sont cependant équivalentes à une perte polynomiale près [10] - même quadratique le plus souvent.

Grâce à la théorie de la complexité qui fournie une approximation du nombre d'étapes nécessaires pour exécuter un calcul, il est possible de se faire une idée du temps réel que mettrait un calcul à être exécuté. Si l'on suppose par exemple que la durée d'une étape est de l'ordre de la μs (microseconde), on peut évaluer le temps d'exécution véritable d'un calculateur - humain

10. Citons par exemple les machines de Kolmogorov-Uspensky où le ruban de la MT est remplacé par un ruban en forme de graphe non orienté qui peut se modifier au cours d'un calcul [Kolmogorov and Uspensky, 1958] ; ou les machines de Shönhage dont le ruban est un graphe orienté que l'on peut modifier à volonté selon certaines règles [Shönhage, 1980].

ou non - qui opère une instruction par μs. Le tableau ci-dessous expose en particulier le temps d'exécution de cinq algorithmes selon leur comportement (C) et la taille (T) des données (1 instruction / μs) :

T/ C	$\log n$	n	$n\log n$	n^2	2^n
10	$3\mu s$	$10\mu s$	$30\mu s$	$100\mu s$	$1000\mu s$
100	$7\mu s$	$100\mu s$	$700\mu s$	$1/100s$	10^{14} siècles
1000	$10\mu s$	$1000\mu s$	$1/100s$	$1s$	astronomique
10000	$13\mu s$	$1/100s$	$1/7s$	$1,7mn$	astronomique
100000	$17\mu s$	$1/10s$	$2s$	$2,8h$	astronomique

Le tableau présenté ci-dessus n'est pas suffisant pour définir ce qui n'est pas calculable manuellement car j'ai fait l'hypothèse selon laquelle la vitesse de calcul est de 1 instruction / μs soit 10^6 instructions / s. La question est donc de savoir où se situent les êtres humains et les ordinateurs par rapport à ce tableau.

Dans le domaine de la mesure du calcul, l'imprécision des chiffres est grande car aucune unité absolue et praticable n'est définie. En effet, diverses unités de bases sont utilisées pour mesurer la puissance des ordinateurs : nombre d'opérations binaires par seconde, nombre de bits utiles calculés par seconde, nombre d'instructions exécutées chaque seconde, nombre de cycles par seconde du microprocesseur et nombre d'opérations en virgule flottante - approximations de nombres réels - effectuées par seconde. Ces critères ont conduit par exemple aux unités que sont les MIPS (million d'instructions / s), les giga-FLOPS (milliard d'opérations en virgule flottante / s) et les giga-hertz (milliard de cycle / s). Puisqu'il est délicat de convertir une de ces unités en une autre, j'ai choisi de ne considérer qu'une seule de ces unités afin de comparer - avec une certaine imprécision - les puissances de calcul : le nombre d'instructions par seconde. Ce choix conduit au tableau ci-dessous qui, en une seule échelle, illustre les puissances de calcul de divers mécanismes [Delahaye, 2003].

Instructions / s	**Mécanisme**
10^9	Console de jeu actuelle
De 10^9 à 10^{10}	Ordinateur de bureau
5×10^{12}	*Deep Blue*
De 10^{13} à 10^{19}	Cerveau humain
2×10^{14}	*Earth simulator* de NEC (2003)
16×10^{15}	*Sequoia* d'IBM (2012)

Ce tableau présente les capacités calculatoires de différents ordinateurs ainsi que celles du cerveau humain. Les données qui concernent la puissance de calcul du cerveau humain sont délicates à établir, en particulier dans le cas d'une évaluation en termes d'instructions par seconde. Ces données sont loins d'être précises - entre 10^{13} et 10^{19} - car les diverses études sur la question ne s'accordent ni sur les résultats ni sur les méthodes fournissant ces résultats [11] [Merkle, 1989].

On peut toutefois conclure sur la base des résultats fournis par le tableau précédent que la puissance de calcul des ordinateurs en terme d'efficacité est supérieure à celle du cerveau humain. Voici deux arguments en faveur de cette conclusion :

1. Premièrement, Gordon Moore remarque en 1965 que depuis l'invention du circuit intégré (1958) le nombre de composants pouvant être intégré sur une puce augmente de manière exponentielle [Moore, 1965]. Plus précisément, on observe un doublement annuel par unité de surface du nombre de transistors, composant électronique actif fondamental en électronique. Moore prédit alors en 1975 que le nombre de composants doublerait tous les 18 mois, ce que l'on dénomme abusivement la *loi de Moore* [12]. Par conséquent même si la puissance calculatoire de *Sequoia*

11. Il n'est pas clair en effet que les ordinateurs et le cerveau humain puissent être comparés en fonction de leur puissance de calcul. De plus, il n'est pas pleinement établi que le cerveau humain puisse être considéré comme un mécanisme qui calcule. Sur ces questions voir [Piccinini, 2007] et [Piccinini, 2008b].

12. Notons qu'il ne s'agit pas, bien sûr, d'une loi mathématique ou physique, mais d'une simple règle hypothétique permettant de prédire l'évolution des performances informatiques. De plus, comme le remarque Delahaye, ces performances résultent d'une combinaison de divers facteurs : « les succès des techniques informatiques auprès du public (facteur

n'atteint pas encore 10^{19} instructions par seconde, il est néanmoins fort probable que les futurs ordinateurs possèdent une puissance supérieure à celle du cerveau.

2. Deuxièmement, l'hypothèse selon laquelle les ordinateurs ont une puissance supérieure à celle du cerveau est encore plus justifiée si l'on quantifie cette puissance à partir de la quantité d'espace mémoire disponible par le calculateur. En effet, le cerveau humain possède d'après Landauer une capacité mémorielle de l'ordre de 200 méga-octets à 1 giga-octet [Landauer, 1986]. Cette mémoire est assez faible comparée à celle des ordinateurs qui surpassent le giga-octet depuis 1994 - 100 giga-octet pour un ordinateur en 2003. Cependant, même si la mémoire des ordinateurs est supérieure à celle du cerveau, la quantité d'espace mémoire ne peut pas être le seul facteur lié à la puissance de calcul car l'efficacité calculatoire dépend aussi fortement de l'organisation d'une telle mémoire.

En conclusion, la construction physique d'un ordinateur est nécessaire pour repousser les limites calculatoires de l'être humain qui sont d'un point de vue de l'efficacité - vitesse et mémoire - inférieures à celles des ordinateurs actuels. Plus précisément, la construction des ordinateurs est nécessaire pour exécuter les calculs qui se situent hors de la faisabilité pratique.

D'un point de vue de la faisabilité pratique, l'étude des ordinateurs est ainsi indispensable pour déterminer ce qui est calculable. Les ordinateurs repoussent les limites de l'être humain en fournissant une quantité de mémoire et de vitesse qui lui permet d'exécuter des tâches impossibles à réaliser manuellement. Dit autrement, les ordinateurs dictent à l'être humain ce qu'il peut mémoriser, ce qu'il peut calculer, et chaque nouvelle avancée technologique, chaque ordinateur nouvellement construit élargie ses possibilités. En bref, les ordinateurs définissent les limites de ce qui est calculable.

Si les ordinateurs sont en pratique indispensables pour dépasser les limites

social) et la concurrence entre firmes (facteur économique) déterminent les bénéfices de l'industrie des semi-conducteurs, qui fixent à leur tour l'investissement en recherche de cette industrie (facteur technologique), lui permettant de surmonter les obstacles qu'oppose la réalité matérielle (facteur physique) ≫ [Delahaye, 2002, p. 79].

de la faisabilité pratique, l'analyse faite par Turing en 1936 permet néanmoins d'affirmer que les limites de la faisabilité en principe sont indépendantes des ordinateurs. Cela signifie que même si les ordinateurs sont en pratique indispensables pour calculer ou mémoriser des données, ils sont en principe négligeables lorsque l'on cherche à définir les limites de ce qui est calculable. Voici une explication détaillée de ce dernier point.

1.2.2 La faisabilité en principe est indépendante de la construction des ordinateurs

J'ai montré ci-dessus que les ordinateurs sont nécessaire pour repousser les limites calculatoires de l'être humain, autrement dit pour exécuter des calculs infaisables en pratique. Néanmoins, quel est le rapport entre les ordinateurs et la faisabilité en principe ? Plus particulièrement, existe-t-il - comme dans le cas de la faisabilité pratique - un rapport de nécessité entre les ordinateurs et ce qui est faisable en principe ?

La faisabilité en principe renvoie aux calculs pouvant être effectués manuellement par l'être humain indépendamment des ressources physiques utilisées pour mener à bien son calcul. La question précédente peut donc être reformulée comme ceci : les êtres humains ont-ils besoin des ordinateurs si l'on fait abstraction des ressources physiques liées aux calculs ? La réponse est négative mais pour l'expliquer, il faut en premier lieu comprendre le rapport entre ce qui est calculable en principe par un ordinateur et ce qui est calculable en principe par un être humain. La compréhension de ce rapport demande en particulier de déterminer (1) ce que peut calculer en principe un ordinateur ; et (2) ce que peut calculer en principe un être humain sans l'aide d'un ordinateur.

Tout d'abord, j'ai défini un ordinateur moderne comme étant un mécanisme dont l'une des caractéristiques est de satisfaire la propriété d'universalité. Une machine, qu'elle soit physiquement construite ou théorique, est dite universelle lorsqu'elle calcule les fonctions effectivement calculables, à savoir les fonctions calculables par une MT. Montrer qu'un ordinateur est universel peut se faire de deux manières : soit à partir de son langage de pro-

grammation soit à partir de son modèle théorique. Dans les deux cas l'idée est de montrer d'une part qu'un calcul effectué par une MT peut être simulé par le langage ou le modèle en question, et d'autre part qu'un calcul effectué par le langage ou le modèle peut être simulé par une MT. C'est par exemple cette stratégie qui est utilisée pour montrer que les langages PASCAL et FORTRAN sont universels.

Plus précisément, l'universalité des ordinateurs se démontre en informatique théorique *via* le modèle de calcul nommé URM (*Unlimited Register Machine*) appelé aujourd'hui RAM (*Random Access Memory)* qui calcule les mêmes fonctions que la MT [13] [Shepherdson and Strurgis, 1963]. Schématiquement, un UMR est une machine théorique qui modélise les microprocesseurs des ordinateurs modernes. Cette machine comporte un nombre de registres $R_1, \ldots, R_n$ gardant chacun en mémoire un entier naturel $r_1, \ldots, r_n$. Le programme d'un UMR est une liste finie d'instructions pouvant être de quatre types différents. Au début d'un calcul, l'UMR commence par exécuter l'instruction numéro 1 sur la donnée en entrée qui est un k-uple $(r_1, \ldots, r_k)$ enregistré dans les registres $R_1, \ldots, R_k$. Ensuite la machine exécute une à une les instructions du programme jusqu'au moment où elle ne peut plus exécuter d'instructions. Enfin, la donnée en sortie est le nombre enregistré dans le registe R_1.

La propriété d'universalité attribuée aux ordinateurs atteste de fait que les fonctions calculables par un ordinateur sont les fonctions Turing-calculables. Mais qu'en est-il de l'être humain ? Possède-t-il lui aussi la propriété d'universalité ?

L'analyse de Turing de la notion de calcul apporte une réponse à ces questions. Pour Turing en effet, la MT est une idéalisation d'un calculateur humain travaillant à l'aide de ressources illimitées en suivant les instructions d'une procédure effective :

> Un homme en train de calculer la valeur d'un nombre peut être comparé à *une machine* susceptible de se trouver dans un nombre

13. J'ai jugé utile de présenter dans l'annexe C cette démonstration qui est trop souvent mise de côté et dont l'oubli a pour conséquence de rendre triviale le caractère universel des ordinateurs.

> fini d'états $q_1, q_2, \ldots, q_n$. La machine est alimentée avec une *bande* (analogue au papier qu'utilise l'homme), divisée en sections (appelées *cellules*), dans chacune desquelles peut être inscrit un *symbole*. Dans la case r est inscrit le symbole $S(r)$. A chaque instant il n'y a qu'une seule cellule dans la machine : c'est la *cellule inspectée*, dans laquelle est inscrit le *symbole inspecté*, le seul dont la machine est pour ainsi dire « directement consciente ». À chaque instant, la liste des comportements possibles de la machine est entièrement déterminée par l'ensemble des ses états q_n et le symbole inspecté $S(r)$. Le couple $(q_n, S(r))$ est appelé la *configuration* de la machine, et c'est donc cette configuration qui détermine l'évolution possible de la machine [...] Ce que j'affirme, c'est que ces opérations englobent toutes celles qui peuvent être utilisées pour calculer la valeur d'un nombre [Turing, 1936, p. 117].
>
> Church et Turing avaient tous les deux à l'esprit le calcul tel qu'il est effectué par un être humain abstrait utilisant certaines aides mécaniques (telles que du papier et des feutres). Le mot 'abstrait' indique que l'argument ne fait pas appel à l'existence de limites pratiques sur le temps et l'espace [Gandy, 1980, p. ?].

Turing défend ainsi à partir de son analyse que les fonctions Turing-calculables sont calculables en principe par un être humain travaillant de manière effective. Cette dernière affirmation - qui peut être identifiée avec la direction *facile*[14] de la *thèse de Church-Turing* (TCT) - est démontrable mathématiquement. Un être humain travaillant de manière effective peut ainsi calculer en principe les fonctions Turing-calculables.

Faisons point sur ce qui a été dit. Pour commencer, les ordinateurs actuels peuvent en principe calculer autant de fonctions que les MT, c'est-à-dire exactement l'ensemble des fonctions Turing-calculables. Cette affirmation peut

14. Pour rappel, la TCT comprend deux directions. Une direction *facile* selon laquelle les fonctions calculables par MT sont calculables par procédure effective ; et une direction *difficile* selon laquelle les fonctions calculables par procédure effective sont calculables par MT. Tandis que la direction difficile suscite encore des débats à propos de sa possible démonstration, la direction facile est quant à elle démontrée mathématiquement [Shoenfield, 1967]. Consulter le chapitre 1 pour plus de détails.

être démontrée à partir du modèle thèorique des ordinateurs - l'UMR - qui est universel. Ensuite, l'analyse de Turing conclue que les fonctions Turing-calculables sont calculables par un être humain disposant de ressources non limitées. Autrement dit le calcul des fonctions Turing-calculables est en principe faisable.

Les deux résultats précédents impliquent ainsi que les être humains sont en principe capables de calculer les mêmes fonctions que les ordinateurs ; que les calculs pouvant être exécutés par les ordinateurs sont en principe faisables par l'être humain. Plus précisément, l'être humain peut suivre chaque programme d'un ordinateur, de la donnée en entrée à la donnée en sortie, et parvenir de lui-même au résultat qui est fourni par l'ordinateur. Pour s'en convaincre, il suffit de rappeler une des contraintes inscrites dans la définition d'une procédure effective - et donc d'un programme - stipulant qu'« un être humain doit pouvoir suivre le calcul étape par étape indépendamment des contraintes de temps et d'espaces mémoire » [Copeland, 2002a, p. 1].

Puisque les êtres humains peuvent en principe avoir accès à tous les résultats qui sont fournis par les ordinateurs, l'étude des ordinateurs n'est pas nécessaire pour déterminer les limites de la faisabilité en principe. Plus généralement, l'étude des limites du calculable peut faire abstraction des ordinateurs car ces derniers n'ont pas d'influence sur ce qui peut être en principe calculé. C'est ainsi que les logiciens des années 1930 tels que Turing et Church ont pu fonder une théorie de la calculabilité qui est indépendante des facteurs matériels, c'est-à-dire qui ne prend pas en compte la construction des ordinateurs.

Mais la théorie de la calculabilité fait encore plus : elle définit les limites du calculable en ne faisant plus référence aux types de calculateurs - humains ou machines - traditionnellement différenciés par leurs capacités. En effet, les fonctions en principe calculables par les ordinateurs et les êtres humains sont d'après cette théorie calculables par une MT ; la calculabilité peut donc être étudiée uniquement à partir de cette formalisation. Il est alors possible de définir les limites du calcul sans les reconduire à des distinctions épistémiques - dans le cas présent, une distinction entre ce qui est calculable par l'être humain et ce qui est calculable par l'ordinateur. L'étude du calculateur -

qu'il soit humain ou machine - n'est donc plus pertinente pour définir les limites du calcul.

Dans la section suivante, je vais cependant expliquer pourquoi l'hyper-calcul pourrait replacer les ordinateurs au centre de l'étude des limites du calcul. Je soutiendrai que les HM, contrairement au ordinateurs actuels, ne sont plus négligeables pour calculer. Précisément, la faisabilité en principe - ce que peut calculer un être humain qui n'est pas limité en ressources physiques - ne permet pas d'effectuer les mêmes calculs que ceux exécutés par les HM. Autrement dit, l'être humain est en principe incapable d'hyper-calculer - accéder aux valeurs de fonctions non Turing-calculables - sans l'aide d'une HM physiquement construite. L'étude du calculateur - et en particulier celle d'une HM - est donc nécessaire pour définir les limites du calcul.

2 L'hyper-machine est nécessaire pour hyper-calculer

J'ai montré dans la section précédente que les ordinateurs sont des outils indispensables d'un point de vue de la faisabilité pratique mais qu'ils sont néanmoins négligeables d'un point de vue de la faisabilité en principe. Ces résultats permettent en particulier de comprendre pourquoi il est possible de faire abstraction des ordinateurs et d'utiliser uniquement la MT pour définir des limites du calcul indépendantes des facteurs technologiques - c'est-à-dire qui ne sont pas relatives à une époque donnée.

Cette section a toutefois pour objectif d'expliquer en quoi la construction d'HM - à supposer qu'elle soit possible - permettrait de réfuter la tentative de définir des limites indépendantes de la technologie. Je défendrai plus précisément la thèse selon laquelle la construction d'HM modifie les limites de la faisabilité en principe - contrairement à la construction des ordinateurs actuels.

Mais que signifie *modifier les limites de la faisabilité en principe* ? La faisabilité en principe dénote les calculs pouvant être exécutés par un être humain lorsqu'il n'est pas limité par les ressources physiques nécessaires à

leur exécution. Ainsi, défendre que la construction d'HM modifie les limites de la faisabilité en principe veut dire affirmer qu'il existe une asymétrie entre ce qui est en principe calculable par un être humain et ce qui est calculable par une HM. En particulier, l'être humain n'est selon moi pas capable - même en principe - d'avoir accès aux valeurs de fonctions non Turing-calculables sans l'aide d'une HM physiquement construite. Autrement dit, l'être humain est d'après moi incapable d'hyper-calculer et cela même s'il n'est pas limité par la quantité de ressources physiques dont il peut disposer.

Défendre une telle position nécessite ainsi d'étudier les capacités calculatoires de l'être humain et plus précisément sa capacité à avoir accès aux valeurs de fonctions non Turing-calculables. Le résultat de cette étude aura en effet pour conséquence de rendre la thèse que je défends vraie ou fausse. Si l'être humain est capable d'hyper-calculer alors la construction d'une HM a le même statut que celle des ordinateurs actuels : elle n'a aucune influence sur les limites de ce qui est calculable. Ma thèse est donc fausse. En revanche, si l'être humain n'est pas capable d'hyper-calculer, cela signifie que la construction d'HM est indispensable pour avoir accès aux valeurs de fonctions non Turing-calculables. Ma thèse est alors vraie.

2.1 Le cerveau humain est-il capable d'hyper-calculer ?

Est-ce que l'être humain est capable d'hyper-calculer ? Autrement dit, est-il capable de calculer plus de fonctions que la MT ? Cette question est intimement liée à l'interprétation du rapport entre calcul et cognition et plus particulièrement du rapport entre la cognition et les contraintes imposées au calcul effectif. Ces contraintes, je le rappelle, sont définies par la théorie de la calculabilité comme étant celles caractérisant la notion de procédure effective. En particulier, un calcul C est une procédure effective si et seulement si elle satisfait les contraintes suivantes [Copeland, 2002a] :

1. Un calcul C doit avoir un nombre fini de symboles et d'instructions.
2. C doit contenir un nombre fini d'étapes.
3. C doit pouvoir être exécuté dans un temps fini.
4. C doit pouvoir être exécuté sans l'aide d'aucune machine.

5. C doit pouvoir être exécuté en principe.
6. Un être humain doit pouvoir exécuter chaque étape de C sans ingéniosité ni intelligence.

Déterminer si l'être humain est capable d'hyper-calcul repose directement sur la satisfaction de ces contraintes. En effet, si un calcul doit satisfaire toutes ces contraintes pour être exécuté par un être humain cela signifie que l'être humain est limité aux procédures effectives et donc que ce dernier ne peut pas dépasser les limites définies par la MT. En revanche, s'il existe des calculs pouvant être réalisés par un être humain qui n'ont pas besoin de satisfaire certaines de ces contraintes, cela voudra dire que l'être humain peut hyper-calculer.

La thèse selon laquelle l'être humain est capable d'hyper-calculer est cristallisée par la thèse que j'appelle [15] :

Thèse (Thèse des hyper-cerveaux)
Le cerveau humain - organe qui effectue les calculs - est capable de calculer au moins une fonction non Turing-calculable.

La vérité de la thèse des hyper-cerveaux repose sur l'interprétation que l'on attribue au rapport entre les contraintes de la notion de procédure effective et les limites calculatoires du cerveau humain. Plus précisément, cette thèse est vraie ou fausse suivant la vérité de l'une des deux affirmations suivantes :

1. L'être humain est restreint à certains moyens finis, c'est-à-dire aux contraintes 1 à 6, parce que ces moyens représentent - dans une forme abstraite - les limitations des capacités calculatoires du cerveau humain.
2. L'être humain n'est pas restreint à certains moyens finis, c'est-à-dire aux contraintes 1 à 6, parce que ces moyens ne représentent pas les limitations des capacités calculatoires du cerveau humain.

15. L'appellation *hyper-cerveau* fait référence à un cerveau capable d'hyper-calculer et a pour origine le terme anglais *supermind* introduit par Bringsjord et Zenzen [Bringsjord and Zenzen, 2003]. Même si elles sont énoncées de manières différentes, ces appellations supposent chacune que le cerveau est à l'origine de l'exécution des calculs.

La vérité de la seconde affirmation implique que le cerveau est capable d'hyper-calculer, à savoir qu'il peut en principe accéder aux valeurs de fonctions non Turing-calculables. Si cette seconde affirmation s'avère être correcte, la thèse des hyper-cerveaux est vraie mais celle que je défends dans cette section - à savoir que la construction physique d'une HM est indispensable pour hyper-calculer - est fausse. Pour que ma thèse soit vraie, il faut ainsi que la première affirmation soit correcte ; autrement dit que la thèse des hyper-cerveaux soit fausse.

Le but de la suite de cette section est donc de montrer que la thèse des hyper-cerveaux n'est pas valide. Pour ce faire, ma stratégie consiste à m'opposer aux principaux arguments fournis en faveur de cette thèse. Ces arguments sont au nombre de quatre. Les deux premiers arguments proviennent des mathématiques et sont introduits par Lucas [Lucas, 1961] et Bringsjord [Bringsjord, 1997]. Ces arguments tentent de montrer que le raisonnement humain ne peut pas être formalisé à partir des systèmes formels, c'est-à-dire à partir d'une MT. Les deux derniers arguments sont quant à eux fondés sur la physique et sont défendus par Penrose [Penrose, 1989] et Siegelmann [Siegelmann, 1995]. Ces arguments ont pour but de prouver que certains processus effectués par le cerveau ne peuvent pas être simulés par une MT.

Avant d'évaluer chacun de ces arguments, je veux tout d'abord clarifier la relation entre la thèse des hyper-cerveaux et la thèse de Church-Turing (TCT). Une telle clarification est nécessaire car selon certains auteurs tels que Gandy la thèse des hyper-cerveaux est fausse en vertu de la TCT :

> L'analyse de Turing fait beaucoup que fournir un argument en faveur de la thèse de Church ; *elle prouve un théorème*. En voici une première version : *Toutes les fonctions qui peuvent être calculées par un être humain peuvent être calculées par une machine de Turing* [Gandy, 1988, p. 76].

Par conséquent comment l'être humain peut-il hyper-calculer si la TCT est un théorème ? Je propose dans ce qui suit d'établir que la thèse des hyper-cerveaux n'est pas en opposition avec la TCT et donc que l'énoncé selon lequel l'être humain est capable d'hyper-calcul est logiquement possible.

2.1.1 La possibilité logique des hyper-cerveaux

Même si la thèse des hyper-cerveaux est d'après moi fausse, je pense qu'il est logiquement possible que le cerveau hyper-calcule. Ce point de vue n'est pourtant pas partagé par l'ensemble de la communauté scientifique. Certains chercheurs soutiennent qu'il n'est pas possible pour l'être humain d'hyper-calculer car une telle possibilité rentre en contradiction avec l'une des thèses suivantes :

T1 Les fonctions calculables par l'être humain sont calculables par MT.

T2 Tout ce qui peut être décrit comme une suite d'étapes précises peut être simulé par une MT [Searle, 1997].

T3 Tous les processus mentaux pouvant être définis comme une opération sur des symboles peuvent être simulés par une MT [Fodor, 1981].

Ces trois thèses affirmant que l'être humain n'est pas capable d'hyper-calcul sont généralement justifiées par le fait qu'elles sont des conséquences de la TCT. Ainsi, puisque la TCT est considérée comme vraie, ces quatre thèses doivent elles aussi être considérées comme vraies.

Dans ce qui suit, je m'oppose à chacune de ces thèses en montrant qu'elles ne reposent pas sur des bases assez solides pour remettre en cause la possibilité logique des hyper-cerveaux. Pour ce faire, je montre soit qu'elles ne résultent en aucun cas de la TCT - thèse **T1** - soit qu'elles peuvent être réfutées par un contre-exemple - thèses **T2** et **T3**.

T1 : Les fonctions calculables par l'être humain sont calculables par MT. La thèse **T1** interprète la TCT comme une thèse à propos des limites calculatoires de l'être humain. Plus précisément, elle considère que la TCT - thèse selon laquelle les fonctions effectivement calculables sont calculables par MT - affirme que les fonctions calculables par un être humain sont calculables par MT.

Cette thèse selon laquelle la TCT est un énoncé à propos des capacités calculatoires de l'être humain provient en partie de l'analyse de la notion de calcul faite par Turing :

> Nous pouvons comparer un homme qui calcule un nombre réel à une machine [...] [Turing, 1936, p. 117].

Mais le fait d'interpréter la TCT comme une thèse centrée sur les capacités humaines a été plus particulièrement effectué par Post :

> En réalité le travail déjà fait par Church et les autres [...] cache le fait qu'une découverte fondamentale concernant les limitations de la puissance mathématique de l'Homo Sapiens a été apportée [...] [Post, 1936, p. 291].

> Etablir cette universalité [la thèse de Church-Turing] ne concerne pas la notion de preuve mathématique, mais concerne l'analyse psychologique des processus mentaux impliqués dans les processus mathématiques combinatoires [Post, 1941, p. 394].

D'après moi, l'interprétation de Post est néanmoins en partie erronée. Elle est correcte au sens où l'analyse de Turing de la notion de calcul a en effet pour origine le calcul mathématique tel qu'il est effectué par un être humain. Elle est en revanche fausse car les résultats de Turing ne concernent pas les limitations de la puissance mathématique de l'Homo Sapiens. En particulier, lorsque Turing énonce qu'un homme qui calcule est comparable à une machine, l'homme qu'il considère n'utilise qu'une *quantité restreinte de ses capacités*. Le calculateur de Turing travaille en effet en exécutant

> [...] des tâches administratives, en travaillant à partir de règles fixes, et sans compréhension [Turing, 1945, pp. 38-39].

Cela signifie plus précisément que la TCT affirme uniquement

> [...] qu'une machine de Turing universelle est capable *d'égaler* le comportement d'un mathématicien travaillant avec du papier et des crayons *via* une méthode algorithmique, affirmation considérablement plus faible qui n'exclut pas la possibilité de l'hypercalcul [Copeland and Proudfoot, 1999, p. 80].

Ainsi la TCT n'énonce rien en ce qui concerne le calcul d'un être humain qui travaille à partir de méthodes qui ne sont pas des procédures effectives. Plus encore, la TCT peut selon moi être considérée comme un argument

sérieux en faveur de la possibilité logique des hyper-cerveaux. Elle nous dit en effet que la MT peut modéliser le comportement calculatoire d'un humain qui ne possède aucune intuition, aucun désir, aucune conscience de soi, aucune créativité, aucune compréhension, en bref, aucune des caractéristiques que l'on associe normalement à un être humain. Et si un calcul universel peut être exécuté en termes de comportement par un modèle aussi simple, il serait étonnant que l'être humain - qui n'est pas limité à la manipulation de symboles - ne soit pas capable de dépasser les capacités de la MT.

En résumé, même si du point de vue du calcul effectif les capacités cognitives attribuées au calculateur humain sont pertinentes, du point de vue du calcul en général « les capacités primitives spécifiées par Turing en 1936 n'occupent pas de positions privilégiées » [Copeland, 2002b, p. 462]. En ce sens, l'analyse de Turing de la notion de calcul et la TCT ne sont pas des preuves de l'équivalence calculatoire entre la MT et l'être humain. Cette analyse laisse donc ouverte la possiblité selon laquelle l'être humain pourrait hyper-calculer. Passons à présent à une réfutation des thèses **T2** et **T3**.

T2 : Tout ce qui peut être décrit comme une suite d'étapes précises peut être simulé par une MT. Cette thèse est défendue par Searle :

> Si la question [est-ce que la conscience est calculable ?] revient à se demander " s'il existe un certain niveau de description où les processus conscients et les processus du cerveau liés à ces processus conscients peuvent être simulés [par une machine de Turing] ? " la réponse est trivialement oui. Tout ce qui peut être décrit comme une suite d'étapes précises peut être simulé [par une machine de Turing] [Searle, 1997, p. 87].

Searle affirme ici que les processus du cerveau sont simulables par une MT car la TCT énonce selon lui que tout ce qui peut être décrit par une suite d'étapes précises peut être simulé par une MT. Cette déduction est toutefois érronée puisque la TCT n'affirme pas que tout ce qui peut être décrit par une suite d'étapes précises peut être simulé par MT. Un contre-exemple possible peut être apporté en considérant la fonction diophantienne

d qui est non Turing-calculable [Matiiassevitch, 1995]. Etant donnée une équation diophantienne [16] codée par un entier naturel x,

$$d(x) = \begin{cases} 1 \text{ si } x \text{ a au moins une solution,} \\ 0 \text{ autrement.} \end{cases}$$

Bien que le calcul de cette fonction ne puisse se faire de façon effective, on peut néanmoins décrire ce calcul comme une suite d'étapes précises :

1. Enumérer les équations diophantiennes par ordre croissant en fonction du nombre de symboles [17].
2. Commencer par tester l'équation x_0 sur la donnée 0 puis sur la donnée 1 et ainsi de suite.
3. Tester l'équation x_1 sur la donnée 0 et ainsi de suite.
4. Continuer en testant l'équation x_n après chaque test de l'équation x_{n-1}.
5. Lors des étapes 2, 3 et 4, si s_{x_n} est une solution de l'équation x_n alors le calcul s'arrête.

Bien évidemment, cette méthode n'est pas effective car on ne peut pas savoir si une équation n'a pas de solution - il y aura dans ce cas toujours un nouvel entier à tester. Toutefois, chaque étape de cette méthode est énoncée de manière précise au sens où il ne peut pas y avoir d'ambiguité à propos de son exécution.

T3 : Tous les processus mentaux pouvant être définis comme une opération sur des symboles peuvent être simulés par une MT. La dernière des trois thèses a pour auteur Fodor. Ce dernier soutient à tort que si l'être humain satisfait certaines contraintes inscrites dans la définition de procédure effective, c'est la preuve que l'être humain est limité au calcul des

16. Une équation diophantienne est une équation dont les cœfficients sont des nombres entiers et dont les solutions recherchées sont également entières. Par exemple, l'équation $x^3 + y^3 + z^3 = 0$ est une équation diophantienne.

17. Cette énumération est possible puisque la cardinalité de l'ensemble des équations diophantiennes est dénombrable ; il existe donc une bijection entre cet ensemble et l'ensemble des entiers naturels.

fonctions Turing-calculables. Afin d'illustrer ce point, concentrons-nous sur le passage suivant :

> Bien que les opérations élémentaires de la machine de Turing soient restreintes, les itérations des opérations permettent à la machine d'exécuter tout calcul bien défini sur des symboles discrets [...] Si un processus mental peut être bien défini comme une opération sur des symboles, il existe une machine de Turing capable d'exécuter le calcul [Fodor, 1981, p. 130].

D'après Copeland, cette dernière citation est fausse car si un être humain était capable de calculer à la manière d'une OM, il exécuterait des opérations bien définies sur des symboles discrets tout en calculant une fonction non Turing-calculable [Copeland, 2000a, p. 20]. Précisément, une OM produit en sortie une suite discrète de symboles - ou même un seul symbole - à partir d'une suite discrète de symboles, au moyen d'une procédure étape par étape qui est détaillée par la table d'instructions de la machine [Copeland and Proudfoot, 1999]. Voici un exemple d'une telle procédure :

1. Sur un premier ruban, entrer un nombre n dont on souhaite déterminer $f(n)$. La fonction f est supposée être non Turing-calculable.
2. Le dispositif de mesure - l'oracle - parcourt un second ruban sur lequel est incrit le nombre τ qui réprésente les résultats de f sous la forme d'une suite discrète de symboles et garde en mémoire le symbole numéro n - ce symbole est extrait en n étapes.
3. Enfin, le dispositif imprime ce symbole sur le premier ruban à la place de la donnée en entrée.

Par conséquent, si un être humain était capable d'exécuter cette procédure - procédure qui est un hyper-calcul - il exécuterait un processus qui n'est pas simulable par une MT tout en utilisant des suites de symboles et des opérations discrètes. La thèse selon laquelle tous les processus mentaux pouvant être définis comme une opération sur des symboles peuvent être simulés par une MT est donc fausse.

Même si je viens de montrer que les thèses **T1**, **T2** et **T3** sont fausses, cela ne signifie cependant que l'être humain est capable d'hyper-calculer. En

effet, même si la possibilité logique des hyper-cerveaux n'est pas remise en cause par la TCT, les tentatives qui ont été proposées afin de prouver que l'être humain peut hyper-calculer ne sont pas exemptes de sérieuses critiques.

Le but de la suite de cette section est de montrer que la thèse des hyper-cerveaux n'est pas valide. Pour ce faire, ma stratégie consiste à m'opposer aux principaux arguments fournis en faveur de cette thèse. Ces arguments sont au nombre de quatre : les deux premiers proviennent des mathématiques et les deux suivants sont fondés sur la physique. Pour chacun de ces arguments, je présente dans un premier temps l'argument tel qu'il est énoncé par son auteur et j'explique dans un second temps pourquoi il ne parvient pas à convaincre clairement que l'être humain est capable d'hyper-calculer.

2.2 Le potentiel de la pensée mathématique

Le premier argument que je souhaite évaluer est proposé par Lucas [Lucas, 1961]. Cet argument fait appel aux théorèmes d'incomplétude de Gödel dans le but de montrer que le potentiel de la pensée mathématique de l'être humain ne peut pas être circonscrit par une MT ou plus précisément par un système formel.

Dans ce qui suit, j'explique l'argument de Lucas afin d'en présenter les principales faiblesses. Pour ce faire, je commence par présenter les notions principales qui sont au cœur de l'argument, à savoir les théorèmes d'incomplétude de Godël et l'équivalence entre les procédures effectives et les systèmes formels. J'analyse ensuite l'argument de Lucas proprement dit en m'appuyant principalement sur son article de 1961. J'expose enfin certaines critiques allant à son encontre.

2.2.1 Les théorèmes d'incomplétude de Gödel

Pour comprendre l'argument de Lucas qui repose majoritairement sur le premier théorème d'incomplétude de Gödel, une présentation des deux théorèmes d'incomplétude est nécessaire. J'ai cependant choisi de présenter informellement ces théorèmes car je pense qu'une compréhension intuitive de ces derniers est suffisante pour saisir de quelle manière ils s'insèrent dans

l'argument de Lucas [18].

Les théorèmes d'incomplétude de Gödel qui sont au nombre de deux concernent la notion de système axiomatique [19] [Gödel, 1931]. Commençons par quelques définitions.

Une *spécification d'un système axiomatique* S est une spécification des éléments qui composent tout système axiomatique : un langage formel L, un ensemble d'axiomes et un ensemble de règles d'inférence. Le *langage* L est formé à partir de symboles de base et de suites finies de symboles de base que l'on nomme *expressions de* L. Il doit en particulier exister une procédure effective permettant de décider si une suite finie de symboles de base est une expression de L. Les *énoncés* ou *formules closes* de L sont sélectionnées à partir des expressions de L et sont produites par une procédure effective à partir des relations de base et des opérations logiques de L. Si A est un énoncé de L et qu'une interprétation des relations de base de L est donnée dans un certain domaine d'objects D, alors A est vrai ou faux sous cette interprétation. Les *axiomes* du système S sont des énoncés de L et les règles d'inférence conduisent à partir de ces axiomes à de nouveaux énoncés ; ici encore, il doit exister une procédure effective pouvant vérifier que les axiomes conduisent à un énoncé donné. Un énoncé de L est *prouvable dans* S si c'est le dernier énoncé d'une *preuve de* S, à savoir d'une suite finie d'énoncés où chacun de ces énoncés est soit un axiome soit un énoncé qui découle d'énoncés précédents *via* les règles d'inférence. Enfin, S est *consistant* s'il n'existe aucun énoncé A de L tel que A et sa négation - notée $\neg A$ - ne sont prouvables dans S.

Hilbert a introduit le terme *métamathématique* pour désigner l'étude des systèmes axiomatiques présents dans de nombreux domaines des mathématiques. En particulier, il propose au début du XX^{e} siècle un large programme - appelé aujourd'hui *programme de Hilbert* - qui consiste à prouver la consistance des systèmes axiomatiques en utilisant uniquement des mathématiques *finitistes*.

18. Consulter [Nagel and Newman, 2008] pour une présentation complète mais informelle des théorèmes d'incomplétude de Gödel. Pour une présentation mathématique, voir [Franzen, 2005] et [Smith, 2007].

19. Pour une analyse de la notion de système axiomatique, voir [Blanché, 1999].

Intuitivement, Hilbert qualifie de finitiste la partie des mathématiques dans laquelle on ne trouve que des propositions qui ne portent pas sur l'infini. Plus précisément, il caractérise le domaine du raisonnement finitiste de la manière suivante :

> Pour qu'on puisse, en effet, appliquer les formes logiques de raisonnement, et effectuer des opérations logiques, une condition préalable et qu'il y ait déjà quelque chose de donné dans la représentation, à savoir de certains objets concrets, extra-logiques, présents à l'intuition et immédiatement perçus antérieurement à toute pensée. Si le raisonnement logique doit donner la certitude, il faut que ces objet se laissent embrasser parfaitement et dans toutes leurs parties, et que leurs propriétés, leurs différences mutuelles, leurs ordonnance à la suite ou à côté l'un de l'autre, soient immédiatement donnés à l'intuition, en même temps que les objets eux-mêmes, comme quelque chose d'irréductible et qui n'a d'ailleurs pas besoin de réduction. Voilà le postulat philosophique fondamental, la condition nécessaire, à mon avis, pour les mathématiques aussi bien pour toute pensée, toute connaissance, tout procédé scientifiques. Et dans les mathématiques, en particulier, les objets de notre étude sont les signes concrets eux-mêmes, dont la figure, conformément à notre condition, et immédiatement perceptible et reconnaissable [Hilbert, 1926, p. 376].

Le qualificatif *finitiste* ne doit cependant pas s'appliquer uniquement aux raisonnements mais aussi aux objets, aux expressions, aux opérations et aux preuves [20].

C'est en étudiant un système formel particulier PM - issu des *Principia Mathematica* [Whitehead and Russel, 1927] - que Gödel a exhibé deux obstacles fondamentaux au programme de Hilbert [Gödel, 1931]. Ces obstacles sont énoncés sous la forme de deux théorèmes :

Théorème (Premier théorème d'incomplétude)

On désigne par PA les axiomes de Peano. Si S est un système formel qui est

20. Le programme de Hilbert est détaillé rigoureusement dans [Dubucs, 1984] et [Zach, 2003].

une extension consistante de PA alors il existe un énoncé arithmétique G qui est vrai mais qui n'est pas prouvable dans S. La vérité d'un énoncé fait ici référence à l'interprétation standard du langage PA dans $\mathbb{N}$.

Théorème (Second théorème d'incomplétude)
Si S est un système formel qui contient PA soit de manière directe ou bien de manière indirecte - comme dans le cas de la théorie des ensembles - et si S est consistant, alors la consistance de S ne peut pas être prouvée dans S, c'est-à-dire qu'elle ne peut pas être prouvée à partir de moyens internes à S.

Le premier théorème d'incomplétude de Gödel montre qu'il n'y a pas de système formel qui englobe toutes les preuves de l'arithmétique. Il n'y a donc pas de théorie complète de toutes les mathématiques. Plus précisément, ce théorème montre que dans tout système formel consistant qui contient l'arithmétique de Peano il existe une formule finitiste G telle que ni G ni $\neg G$ ne sont des théorèmes du système. On peut revanche démontrer dans les mathématiques que la proposition qui correspond à G est vraie. Une telle formule est appelée un *énoncé de Gödel* et exprime le fait suivant : G n'est pas prouvable dans S.

Le second théorème d'incomplétude de Gödel montre que la consistance d'un système formel consistant qui contient l'arithmétique de Peano ne peut pas être démontrée par des moyens qui seraient codifiés dans l'arithmétique de Peano ; *a fortiori*, on ne peut démontrer cette consistance par des moyens finitistes qui seraient codifiés dans l'arithmétique de Peano. Cela n'exclut pas que l'on puisse démontrer cette consistance par d'autres moyens, qui doivent dépasser le cadre de l'arithmétique de Peano.

L'argument de Lucas utilise en particulier le premier théorème d'incomplétude pour montrer que le potentiel de la pensée mathématique ne peut pas être circonscrit par une MT, c'est-à-dire par un système formel. Toutefois, la compréhension de l'argument de Lucas demande d'éclaircir un dernier point qui concerne l'équivalence entre les notions de MT et de système formel. En effet, le premier théorème d'incomplétude traite de la notion de système formel et non de MT.

Le lien qui existe entre système formel et procédure effective est très fort car il est caractérisé par une équivalence entre ces deux notions :

> Le travail de Turing donne une analyse du concept de "procédure mécanique" (alias "algorithme" ou "procédure de calcul" ou "procédure combinatoire finie"). Ce concept est démontré comme étant équivalent avec celui de "machine de Turing". Un système formel peut simplement être défini comme une procédure mécanique pour produire des formules, appelées formules prouvables. Pour tout système il en existe une [procédure mécanique][...] qui possède les même formules prouvables (et vice versa)[...] [Gödel, 1965] dans [Gödel, 1986, p. 369].

En d'autres termes, Gödel identifie la notion de système formel avec celle de procédure mécanique. Il démontre pour cela les deux propositions suivantes :

Proposition
Pour tout système formel, il existe une procédure mécanique associée à ce système qui peut produire ses formules prouvables.

Proposition
Toute procédure mécanique peut être effectivement transformée en une autre procédure mécanique qui peut produire les formules prouvables d'un système formel.

La compréhension de l'équivalence entre les systèmes formels et les procédures effectives - ou ce qui revient au même, entre les systèmes formels et les MT - conjuguée à celle des théorèmes d'incomplétude de Gödel, permet à présent de présenter l'argument de Lucas en faveur de la thèse des hyper-cerveaux.

2.2.2 L'argument de Lucas

Dans son article de 1961, Lucas utilise le premier théorème de Gödel pour formuler un argument en faveur de la thèse des hyper-cerveaux [Lucas, 1961].

Plus précisément, Lucas est contre une *thèse mécaniste* affirmant que l'esprit humain [21] peut être *expliqué* à partir de la notion de machine :

> Les théorèmes de Gödel sont pour moi la preuve que le mécanisme est faux, c'est-à-dire, que les êtres humains ne peuvent pas être expliqués en termes de machines [Lucas, 1961, p.112].

La thèse mécaniste que veut réfuter Lucas est néanmoins trop imprécise car il existe une myriade de thèses dites *mécanistes* au sein de la littérature [22]. C'est pour cela que Feferman précise la thèse combattue par Lucas de la façon suivante [Feferman, 2011] :

Thèse (Thèse mécaniste)
Dans la mesure où la pensée mathématique est concernée, le cerveau est mécanique au sens où l'ensemble de tous les théorèmes mathématiques, actuel ou potentiel, produit par le cerveau est l'ensemble des énoncés pouvant être produit par un système formel.

Pour montrer que la thèse mécaniste est fausse, Lucas procède comme suit :

> [Nous] construisons maintenant une formule de Gödel [l'énoncé G utilisé dans le premier théorème d'incomplétude] dans ce système formel. Cette formule ne peut pas être *prouvée-dans-le-système*. Par conséquent la machine ne peut pas produire la formule correspondante à G comme étant vraie. Mais nous pouvons voir que la formule de Gödel est vraie : n'importe quel être rationel pourrait suivre l'argumentation de Gödel, et se convaincre que la formule de Gödel, bien qu'étant improuvable-dans-le-système, est néanmoins [...] vraie [...] Cela montre que la machine ne peut pas être un modèle complet et adéquat de l'esprit humain. Il ne peut pas faire *tout* ce que l'esprit peut faire, puisque bien qu'il

21. Dans son article, Lucas utilise le terme *mind* que j'ai traduit par *esprit humain*. J'utiliserai donc dans ce qui suit le terme *esprit humain* à la place de *cerveau humain*.

22. Comme l'explique Bringsjord, la doctrine mécaniste est très vague car elle peut renvoyer entre autres aux thèses suivantes : (1) penser c'est calculer ; (2) les êtres humains sont des MT ; (3) les êtres humains se comportent comme des ordinateurs ; ou (4) l'esprit humain ne calcule que des fonctions Turing-calculables [Bringsjord et al., 2011].

> puisse faire beaucoup de choses, il y a toujours quelque chose qu'il ne peut pas faire, mais qui peut être fait par l'esprit [...] Par conséquent nous ne pouvons jamais espérer produire une machine qui est capable de faire tout ce qu'un esprit humain [Lucas, 1961, p.115].

L'argument de Lucas est simple à résumer. D'après le théorème de Gödel, pour tout système formel consistant et assez puissant pour produire de l'arithmétique simple, il existe une formule de Gödel G vraie qui est *indécidable*, au sens où ni elle ni sa négation ne peut être prouvée par le système. Puisque G est vraie mais est indécidable, cette dernière échappe au pouvoir déductif du système formel. Cependant, G n'échappe pas à la pensée mathématique humaine car le mathématicien sait que G est vraie. En d'autres termes, l'être humain peut faire quelque chose de plus qu'un système formel ou qu'une MT, à savoir affirmer que G est vrai.

D'après Lucas, la thèse mécaniste est fausse puisque le cerveau peut produire un théorème d'un système formel S, à savoir que G est un théorème de S, qui ne peut pas être produit par S lui-même. Plus encore, même si l'on rajoute la formule G à l'ensemble des axiomes de S, la procédure de Gödel permet d'exhiber une nouvelle formule G' indécidable dans S mais dont on peut affirmer la vérité. En fait, quelle que soit la complexité du système formel que l'on considère, ce système restera par définition un système formel qui est assujetti à la procédure de Gödel. Pour résumer,

> Nous essayons de produire un modèle de l'esprit qui soit mécanique (essentiellement *mort*), mais l'esprit humain est en fait *vivant* et peut toujours faire un peu mieux qu'un système formel, ossifié, mort. Grâce au théorème de Gödel, le cerveau a toujours le dernier mot [Lucas, 1961, p. 116].

2.2.3 Objections

Malgré les nombreuses objections qui ont été énoncées à son encontre et l'ancienneté de sa formulation (1961), l'argument de Lucas se situe encore au

centre du débat contemporain sur les limitations de l'esprit humain [23]. Loin de présenter l'ensemble de ces objections, je me limite à deux objections qui sont pour moi assez puissantes pour réfuter la position de Lucas selon laquelle le théorème d'incomplétude de Gödel permet de montrer que l'esprit humain ne peut pas être réduit à une MT.

Première objection. Cette objection repose sur la consistance de l'esprit humain qui est une condition nécessaire pour que le raisonnement de Lucas soit vrai. Expliquons tout d'abord pourquoi la consistance de l'esprit humain est une condition nécessaire au bon déroulement de son raisonnement.

Le cœur du raisonnement de Lucas est de dire que l'esprit humain ne peut pas être formalisé par un système formel. En particulier, Lucas s'appuie sur une démonstration par l'absurde. Supposons que l'esprit humain est formalisé par un tel système. D'après le théorème d'incomplétude de Gödel, il existe une formule G qui ne peut pas être prouvée par le système mais dont on peut établir qu'elle est vraie. Par conséquent, puisque les limitations des systèmes formels ne sont pas identiques à celles des êtres humains il suit que l'esprit humain n'est pas un système formel.

Toutefois, la conclusion de Lucas n'est vraie que si le système formel est consistant. La raison en est que le premier théorème de Gödel s'énonce sous la forme d'une implication dont l'antécédant requiert la consistance du système : *si un système formel est consistant* et qu'il contient les axiomes de Peano alors il existe une formule indécidable G. Plus précisément, la consistance du système est une condition nécessaire au théorème de Gödel car si le système est inconsistant alors la formule G ainsi que sa négation deviennent des théorèmes du système. Un système formel est consistant - ou non contradictoire - s'il n'existe pas d'énoncé E tels que E et $\neg E$ sont prouvables dans le système. Autrement dit, un système formel est consistant si la disjonction exclusive suivante est vraie : pour tout énoncé E, soit E est prouvable dans le système soit $\neg E$ est prouvable dans le système. Ainsi dans le cas où le

23. Le débat est actuellement repris sous une forme légèrement différente par Penrose, Shapiro et Lindström [Penrose, 1994], [Shapiro, 2003], [Lindström, 2001], [Lindström, 2006].

système est inconsistant, tout énoncé est prouvable dans le système et le théorème de Gödel ne peut pas s'appliquer à ce système. Pour résumer, le raisonnement de Lucas repose sur l'hypothèse selon laquelle l'esprit humain est consistant.

La première objection allant à l'encontre de l'argumentation de Lucas est de dire que même si l'esprit humain est consistant, ce dernier ne peut pas établir sa propre consistance. Cette affirmation est défendue par Putnam à partir du second théorème d'incomplétude de Gödel [Putnam, 1960]. D'après Putnam, même si l'être humain est consistant ce dernier ne peut pas le prouver à cause du second théorème d'incomplétude. Ce théorème énonce en effet qu'il est impossible de prouver la consistance d'un système formel à l'intérieur du système proprement dit, ce qui est une conséquence directe du premier théorème d'incomplétude.

Lucas défend tout de même son raisonnement initial en soulignant que Gödel n'a pas apporté la preuve selon laquelle la consistance d'un système formel est impossible, mais la preuve selon laquelle la consistance d'un système formel ne peut pas être démontrée par le système lui-même [Lucas, 1990]. Il existe en effet des preuves finitistes de la consistance de la logique propositionnelle ainsi que de la logique des prédicats du premier ordre ; il existe aussi une preuve non finitiste de la consistance de l'arithmétique de Peano [Gentzen, 1969]. Même si la consistance d'un système ne peut pas être démontrée au sein du système lui-même, il peut ainsi exister des moyens extérieurs pouvant être utilisés afin de la démontrer. Autrement dit, même si l'esprit humain ne peut pas établir sa propre consistance, des moyens extérieurs à l'esprit humain pourraient en être capable. Lucas était cependant conscient de la fragilité de sa réponse puisqu'un système extérieur à l'esprit humain qui fournirait une preuve de consistance pour l'esprit humain devrait à son tour être consistant, ce qui demanderait une preuve provenant d'un autre système, et ainsi de suite [Lucas, 1996]. C'est pourquoi Lucas en est venu à considérer la consistance de l'esprit humain non pas comme une affirmation devant être démontrée mais comme une hypothèse nécessaire à la pensée :

> [...] Nous devons supposer notre propre consistance pour que la pensée soit possible. Une telle supposition n'est pas [...] quelque

> chose pouvant être soigneusement établi à partir d'un enchaînement d'arguments. Au contraire, une telle supposition est une hypothèse nécessaire que nous devons faire si nous voulons commencer à penser [Lucas, 1976, p. 147].

La proposition que fait Lucas de considérer la consistance de l'esprit humain comme une hypothèse nécessaire à la pensée n'est pas une solution à l'objection de Putnam et pourrait même être considérée comme un aveu de faiblesse de sa part. Plus grave encore, l'objection de Putnam n'est pas la seule objection qui ébranle l'argumentation de Lucas.

Seconde objection. La seconde objection ne s'appuie pas sur la consistance de l'esprit humain mais sur l'énoncé de Gödel G qui permet selon Lucas de différencier l'esprit humain de la machine. Cette objection possède plus précisément trois formulations différentes.

La première formulation de l'objection est énoncée par Benacerraf [Benacerraf, 1967]. Pour ce dernier, l'esprit humain est tellement complexe que le raisonnement de Lucas - selon lequel si l'esprit humain est formalisé par un système formel alors il pourrait affirmer qu'un énoncé improuvable dans le système est néanmoins vrai - ne pourra jamais être vérifié. D'après Benacerraf, la construction d'un énoncé de Gödel nécessite une solide compréhension du système de déduction qui est à l'œuvre dans le système. Par exemple, les mathématiciens comprennent suffisamment le système des *principia* pour qu'il soit possible de construire un énoncé de Gödel. En revanche, l'extrême complexité du système de déduction à l'œuvre dans l'esprit humain rend la construction d'un énoncé de Gödel très improbable. Comme l'affirme Benacerraf, l'esprit humain serait dans ce cas une machine de Turing très complexe mais une machine de Turing tout de même. Lucas répond à Benacerraf que l'être humain pourrait être aidé par d'autres mathématiciens afin de produire un énoncé de Gödel [Lucas, 1996]. Autrement dit, même si la construction d'un tel énoncé semble être difficile, cette construction pourrait être menée à bien avec une aide extérieure.

L'objection de Benacerraf est néanmoins reprise par Lewis mais sous une forme différente [Lewis, 1969]. Pour reprendre les mots de Lucas,

> D'après Lewis, j'ai montré qu'il existait une certaine *arithmétique de Lucas* qui était clairement vraie mais qui ne pouvait pas être produite par une machine de Turing. Si je pouvais produire l'intégralité de l'arithmétique de Lucas, alors je ne serais certainement pas une machine de Turing. Mais il n'y a en général aucune raison de supposer que je suis capable de produire l'ensemble des théorèmes de l'arithmétique de Lucas [Lucas, 1970, p. 149].

Pour Lewis, l'esprit humain est sujet aux mêmes limitations que celles attribuées par les théorèmes de Gödel aux systèmes formels. Plus précisément, l'arithmétique de Peano est l'arithmétique que les machines - ou systèmes formels - peuvent produire, tandis que l'arithmétique de Lucas est d'après Lewis l'arithmétique que les humains peuvent produire. Pour Lucas, l'arithmétique de Lucas n'est pas limité aux même titre que l'arithmétique de Peano car les énoncés de Gödel sont indécidables pour l'arithmétique de Peano tandis qu'ils sont décidables pour l'arithmétique de Lucas. Autrement dit, les énoncés de Gödel font partie de l'ensemble des théorèmes que peut prouver l'arithmétique de Lucas ; ce qui implique selon Lucas que les humains ne sont pas des machines. Lewis affirme quant à lui que Lucas n'a jamais montré qu'un être humain peut produire l'arithmétique de Lucas dans son intégralité, condition nécessaire pour produire les énoncés de Gödel en question.

Ici encore, Lucas répond à l'objection de Lewis en faisant remarquer qu'il n'est pas nécessaire de produire l'intégralité de l'arithmétique de Lucas pour que son raisonnement soit juste. En effet, il suffit que l'être humain puisse produire un seul théorème considéré comme vrai qui ne peut pas être produit par une machine :

> Ce que je dois faire consiste à montrer qu'un esprit humain peut produire, non pas l'intégralité des théorèmes de l'arithmétique de Lucas, mais seulement une petite partie de cette arithmétique. Et cela je pense que je peux le faire, merci au théorème de Gödel [Lucas, 1970, p. 149].

Cette réponse faite par Lucas permet de minimiser l'impact que pourrait avoir l'objection de Lewis. Plus généralement, même si les objections de

Benacerraf et de Lewis mettent en lumière certaines failles au sein de l'argumentation de Lucas, elles ne parviennent pas rejeter l'argument de Lucas. Il existe cependant une dernière formulation de l'objection qui est d'après moi imparable.

L'objection en question est due à Whiteley et consiste à dire que l'esprit humain possède les mêmes limitations que celles attribuées par Lucas aux systèmes formels [Whiteley, 1962]. Plus précisément, Whiteley affirme qu'il existe des énoncés indécidables pour l'esprit humain tel que l'énoncé suivant appelé *énoncé de Whiteley* : *Lucas ne peut pas affirmer que l'énoncé de Whiteley est vrai tout en étant consistant*. D'une part, si l'énoncé de Whiteley est vrai alors Lucas est inconsistant - je rappelle que l'esprit humain doit être consistant pour que le raisonnement de Lucas soit valide. D'autre part, si l'énoncé de Whiteley est faux alors sa négation - *Lucas peut affirmer que l'énoncé de Whiteley est vrai tout en étant consistant* - est vraie ce qui implique que l'énoncé de Whiteley est vrai ; contradiction. L'énoncé de Whiteley doit être ainsi considéré comme une authentique preuve énonçant que le théorème de Gödel ne peut pas être utilisé pour montrer que l'esprit humain est supérieur aux systèmes formels. En d'autres termes, l'énoncé de Whiteley est aux esprits humains ce que l'énoncé de Gödel est aux systèmes formels.

Malgré l'échec de l'argument de Lucas, d'autres tentatives fondées sur la logique ont été entreprises. Il existe en particulier un argument différent de celui de Lucas ayant pour but de montrer que l'esprit ne peut pas être réduit à une MT. Cet argument introduit par Bringsjord ne repose toutefois pas sur un théorème logique particulier tel que le théorème de Gödel, mais sur le raisonnement logique en général.

2.3 Les raisonnements logiques et la logique infinitaire

2.3.1 L'argument de Bringsjord

L'argument de Bringsjord provient de la logique infinitaire et a pour but de montrer que la nature infinitaire du raisonnement mathématique est une preuve de la vérité de la thèse des hyper-cerveaux [Bringsjord, 1997]. Plus

précisément, Bringsjord veut montrer que l'être humain est capable de faire des raisonnements qui ne peuvent pas être formalisés par la logique du premier ordre, c'est-à-dire qui ne peuvent pas être simulés par une MT. Voici l'argumentation étape par étape :

1. Tous les raisonnements sont simulables par MT [hypothèse de départ que Bringsjord souhaite contredire].
2. Pour chaque raisonnement R il existe une MT M et un calcul C de M tel que $R = C$ [de 1].
3. Pour chaque calcul C de chaque MT M, il existe une déduction équivalente D à l'intérieur d'une certaine instanciation du système de premier ordre $\mathcal{L}_I$ [d'après la thèse de Church-Turing].
4. Pour chaque raisonnement R, il existe une déduction D à l'intérieur d'une certaine instanciation du système de premier ordre $\mathcal{L}_I$ tel que $R = D$ [de 2, 3].
5. Il existe un raisonnement E^* à partir de $\mathcal{L}_{w_1 w}$ tel que pour toute déduction D à l'intérieur d'une certaine instanciation du système de premier ordre $\mathcal{L}_I$ $E^* \neq D$.
6. Il est faux que tous les raisonnements sont simulables par MT [*reductio ad absurdum* à partir de 4 et 5].

Pour comprendre correctement ce raisonnement, il est nécessaire de détailler l'étape 5 qui est décisive pour l'argumentation. Pour Bringsjord, la meilleure façon de montrer que le raisonnement mathématique n'est pas simulable par MT consiste à rendre explicite la notion de raisonnement infini. L'auteur se concentre lors de son exposition sur un système infinitaire particulier appelé $\mathcal{L}_{w_1 w}$. Ce système est fondé sur la logique du premier ordre et a pour principal intérêt de pouvoir exprimer des concepts mathématiques qui ne peuvent pas être exprimés en logique du premier ordre - un système généralement appelé $\mathcal{L}_I$ [Keisler, 1971]. En particulier, les concepts d'*interprétation finie* et d'*arithmétique ordinaire* ne peuvent pas être exprimés en logique du premier ordre mais peuvent être exprimés dans $\mathcal{L}_{w_1 w}$. J'explique ci-dessous uniquement le premier concept - pour le second concept, voir [Bringsjord, 1997, p. 12].

Dans le but de comprendre qu'il n'est pas possible d'exprimer la notion ordinaire de finitude en logique du premier ordre, considérons le $\mathcal{L}_I$-énoncé suivant

$$\psi_{\geq 2} = \exists x \exists y \; x \neq y$$

Puisque $\psi_{\geq 2}$ exprime qu'*il existe deux éléments distincts x et y*, toute interprétation pour laquelle $\psi_{\geq 2}$ est vraie doit posséder un domaine qui contient au moins 2 éléments. Cette formule peut évidemment être étendue à 3 éléments par la formule

$$\psi_{\geq 3} = \exists x \exists y \exists z \; x \neq y \wedge x \neq z \wedge y \neq z$$

qui est vraie si et seulement si le domaine contient au moins trois éléments. Maintenant, quelle est la procédure si l'on veut exprimer qu'une interprétation est finie, autrement dit que son domaine contient un nombre fini d'éléments ? Une procédure possible consiste à regrouper l'ensemble de toutes les formules qui expriment *il existe n éléments dans le domaine* pour tout $n \in \mathbb{N}$ et $n \geq 2$. Formellement un tel ensemble nommé Ω est

$$\Omega = \{\psi_{\geq 2} / n \geq 2\}$$

L'ensemble Ω soulève néanmoins le problème suivant. Puisque toute interprétation pour laquelle chaque membre de Ω est vrai doit être une interprétation qui contient au moins 2 éléments, 3 éléments, 4 éléments, *ad infinitum*, il suit que cette interprétation possède un domaine contenant un nombre infini d'éléments. Par conséquent la procédure, loin d'exprimer en logique du premier ordre la notion d'interprétation finie, exprime celle d'interprétation infinie. Plus généralement, il n'existe pas d'ensemble de formules de la logique du premier ordre pouvant exprimer le concept d'interprétation finie.

C'est en partie à cause de telles limites que les logiciens ont étudié des systèmes comme $\mathcal{L}_{w_1 w}$ qui peuvent exprimer des énoncés non exprimables en logique du premier ordre. L'idée de base qui est derrière $\mathcal{L}_{w_1 w}$ est très

simple. Contrairement à $\mathcal{L}_I$, ce système autorise les conjonctions et les disjonctions infinies dont le nombre d'éléments n'est pas supérieur à celui de $\mathbb{N}$ - je dénote ce nombre par w. Cette idée fondamentale est formalisée en ajoutant à l'alphabet habituel de la logique du premier ordre les symboles $\bigwedge$ et $\bigvee$ qui dénotent respectivement la conjonction et la disjonction infinie. De plus, on ajoute aux règles de formation des formules de la logique du premier ordre la règle suivante :

- Si Φ est un ensemble de formules bien formées $\{\varphi_1, \varphi_2, \ldots\}$ dont le nombre de formules n'est pas supérieure à w, alors $\bigwedge \Phi$ et $\bigvee \Phi$ sont aussi des formules bien formées.

De même, la notion de formule infinie vraie dans une interprétation est fixée en généralisant le notion de vérité telle qu'elle est définie en logique du premier ordre :

1. Une disjonction infinie $\bigvee \Phi$ est vraie dans une interprétation $\mathcal{I}$ si et seulement s'il existe une formule ϕ dans Φ qui est vraie dans $\mathcal{I}$.
2. Une conjonction infinie $\bigwedge \Phi$ est vraie dans une interprétation $\mathcal{I}$ si et seulement si toutes formules ϕ dans Φ est vraie dans $\mathcal{I}$.

Enfin, les preuves dans $\mathcal{L}_{w_1 w}$ peuvent être infiniment longues [Barwise, 1969], [Ebbinghaus et al., 1984].

Une des propriétés remarquables de la logique $\mathcal{L}_{w_1 w}$ est qu'elle permet de surmonter la limitation de la logique du premier ordre énoncée plus haut, à savoir que le concept de finitude ne peut pas être exprimé par un ensemble de $\mathcal{L}_I$-formules. En particulier, il existe une $\mathcal{L}_{w_1 w}$-formule qui exprime le concept de finitude où toute interprétation satisfaisant cette formule possède un domaine consitué d'un nombre fini d'éléments :

$$\bigvee_{n<w} \exists x_1 \ldots \exists x_n \forall y (y = x_1 \vee \ldots \vee y = x_n)$$

Cette formule est une disjonction infinie où chaque disjoint possède une valeur différente de n. En particulier, (1) toute interprétation dont le domaine est fini satisfait cette formule pour un de ses disjoints - pour un certain n ; et

(2) toute interprétation qui satisfait cette formule pour un de ses disjoints - pour un certain n - possède nécessairement un domaine fini.

Ce dernier résultat permet à Bringsjord de conclure que le raisonnement mathématique qui porte sur $\mathcal{L}_{w_1w}$ ne peut pas être traduit en logique du premier ordre. Il existe ainsi un raisonnement E^* à partir de $\mathcal{L}_{w_1w}$ tel que pour toute déduction D à l'intérieur du système de premier ordre $\mathcal{L}_I$, $E^* \neq D$.

Pour résumer, l'argument de Bringsjord est de dire que puisque l'être humain peut faire des raisonnements portant sur $\mathcal{L}_{w_1w}$ qui est une logique pouvant exprimer des concepts ne pouvant pas être exprimés par la logique du premier, et que la MT a le même pouvoir expressif que la logique du premier ordre, l'être humain est par conséquent supérieur à la MT.

Deux objections ont toutefois été adressées à l'encontre de l'argument de Bringsjord. En ce qui concerne le premier argument, je prends position en faveur de Bringsjord car je pense que cet argument n'est pas assez solide pour renverser son raisonnement. Je pense en revanche que la seconde objection soulève de réels problèmes à propos de l'argument de Bringsjord.

2.3.2 Objections

Objection 1. La première objection que l'on peut énoncer à l'encontre de l'argument de Bringsjord consiste à dire que les raisonnements mathématiques avec $\mathcal{L}_{w_1w}$ ne sont que des manipulations d'expressions finies pouvant être produites par une MT :

> [...] Clairement les êtres humains ne peuvent pas manipuler une expression infinie. Par conséquent, avoir un 'raisonnement infini' avec $\mathcal{L}_{w_1w}$ veut dire avoir un raisonnement à l'aide d'une manipulation de suites finies de symboles qui sont utilisées pour représenter des expressions infinies hypothétiques. Par exemple, examine la formule que tu as utilisé dans ton raisonnement [24]. Tu noteras que cette formule est une suite finie de symboles qui peut parfaitement être inscrite en une ligne sur une feuille de $8,5$ cm $\times$ 11 cm. Et bien sûr nous savons tous que les machines de

24. A savoir $\bigvee_{n<w} \exists x_1 \ldots \exists x_n \forall y (y = x_1 \vee \ldots \vee y = x_n)$.

> Turing n'ont aucun problème à manipuler de telles suites [...] [Bringsjord, 1997, 17] .

Pour Bringsjord, cette objection reflète une thèse à propos du calcul qui semble être similaire à la thèse finitiste que défend Hilbert en 1926. Comme je l'ai expliqué dans la section précédente, les preuves mathématiques sont pour Hilbert présentées invariablement comme des suites finies de symboles sur des morceaux finis de papier. En partie à partir de ce constat, Hilbert défend la thèse selon laquelle les preuves mathématiques sont des méthodes mécaniques, c'est-à-dire des méthodes de longueurs finies qui peuvent être exécutées étape par étape à partir d'un ensemble fini de règles explicites. Plus précisément, cette thèse est que chaque problème mathématique peut être résolu par une méthode finitiste. Par exemple, démontrer la consistance d'une théorie doit impliquer uniquement des procédures finies faisant références à un nombre fini de propriétés que possèdent les formules de la théorie.

Gödel a toutefois porté un coup fatal au programme de Hilbert en montrant que pour chaque système formel assez puissant pour exprimer l'arithmétique élémentaire, il existe une formule de la forme $\forall x \phi(x)$ à propos des entiers naturels - autrement dit une formule qui exprime que tous les entiers naturels ont une certaine propriété ϕ - qui ne peut pas être prouvée par des moyens finis, même si chaque $\phi(0), \phi(1), \phi(2), \ldots, \phi(n)$, $\phi(n+1), \ldots$ - où chacune de ces formules exprime qu'un entier naturel particulier possède la propriété ϕ - est prouvable par une preuve finie à partir des axiomes énoncés en logique du premier ordre qui caractérisent les entiers naturels. En d'autres termes, Gödel a trouvé une formule qui est vraie pour tous les entiers naturels mais qui ne peut pas être démontrée dans le système à l'aide de moyens finis.

Pour Bringsjord, le résultat de Gödel a poussé la communauté scientifique à suggérer que les formalisations de la logique du premier ordre doivent être remplacées par des formalisations de $\mathcal{L}_{w_1 w}$. Plus précisément, l'objection illustrée par le passage cité plus haut n'est pas pertinente du point de vue de Bringsjord car elle est selon lui une énième tentative en faveur de la nécessité d'un raisonnement mathématique réductible à des méthodes finies ; idée remise en cause par Gôdel et remplacée par celle selon laquelle le raisonnement

doit se faire à partir de méthodes infinitaires fondées sur $\mathcal{L}_{w_1w}$.

En réalité, ce n'est pas si simple. Bien que le système $\mathcal{L}_{w_1w}$ permet d'exprimer des propriétés qui ne peuvent pas être exprimées par $\mathcal{L}_I$, cette logique infinitaire n'est pas exempt de problèmes. En particulier, certains puissants théorèmes tels que le théorème de compacité qui sont prouvables dans $\mathcal{L}_I$ ne peuvent pas être retrouvés dans $\mathcal{L}_{w_1w}$ [Bell, 2006]. Ainsi, la position selon laquelle $\mathcal{L}_I$ doit être remplacée par $\mathcal{L}_{w_1w}$ est loin de faire l'unanimité parmi les logiciens.

Je pense néanmoins que l'objection faite à Bringsjord n'est pas convaincante. En effet, cette objection affirme que le raisonnement sur $\mathcal{L}_{w_1w}$ est simulable par MT puisque les $\mathcal{L}_{w_1w}$-formules peuvent être exprimées à l'aide de suites finies de symboles qui sont manipulables par une MT. Mais prenons la même objection dans le cas de la fonction suivante où x est une équation diophantienne :

$$f(x) = \begin{cases} 1 \text{ si } x \text{ a au moins une solution,} \\ 0 \text{ autrement.} \end{cases}$$

Bien que cette fonction puisse être exprimée à partir d'une suite finie de symboles qui est manipulable par une MT, elle a été démontrée par Matiiassevitch comme étant non Turing-calculable [Matiiassevitch, 1995]. Cela signifie plus précisément que le calcul de cette fonction n'est pas simulable par une MT même si la définition de cette fonction est quant à elle manipulable par une MT. Par conséquent tout ce qui peut être exprimé par une suite finie de symboles n'est pas obligatoirement simulable par une MT. Il existe donc une différence entre ce qui est *manipulable* par une MT et ce qui est *simulable* par une MT. De façon équivalente, il existe une différence entre le fait que l'expression d'un énoncé soit manipulable par une MT et le fait que le raisonnement à propos de cet énoncé soit simulable par une MT. Je ne pense donc pas que l'objection soit suffisante pour remettre en cause l'argument de Bringsjord.

Cette objection n'est cependant pas la seule faite à Bringsjord vis-à-vis de sa conclusion selon laquelle un raisonnement sur $\mathcal{L}_{w_1w}$ est une preuve de la supériorité des êtres humains sur la MT.

Objection 2. La seconde objection peut être énoncée en ces termes :

> [...] Note que [dans ton argumentation] tu passes de 'concevoir, réfléchir sur, raisonner à propos de' (expressions pour lesquelles je n'ai pas de problème) à 'manipuler'. Ici tu passes de la description de suites infinies à l'utilisation de ces mêmes suites. C'est précisément ton erreur [...] Il est sûrement possible pour un mathématicien d'utiliser certaines représentations mentales finies pour raisonner *à propos* de $\mathcal{L}_{w_1w}$. Mais aucun mathématicien n'est capable de raisonner *avec* $\mathcal{L}_{w_1w}$ car ce système ne peut pas être une description adéquate de notre langage de la pensée [Bringsjord, 1997, pp. 22-23].

Bringsjord répond à cette objection que même si les mathématiciens - et les non mathématiciens - ont pour habitude de raisonner *avec* des représentations et des systèmes de raisonnements finis - comme par exemple $\mathcal{L}_I$ - cela ne rend pas impossible le fait de raisonner *avec* $\mathcal{L}_{w_1w}$. Le point important que rappelle l'auteur est qu'il a choisi $\mathcal{L}_{w_1w}$ pour la bonne raison que certains raisonnements *à propos* de ce système logique reviennent clairement à raisonner *avec* un système de représentations et de raisonnements ayant au moins le caractère infinitaire de $\mathcal{L}_{w_1w}$. Pour s'en rendre compte, il est nécessaire de regarder ce qui se passe lorsqu'un théorème à propos de $\mathcal{L}_{w_1w}$ est prouvé. Prenons comme exemple le théorème suivant :

Théorème (Théorème d'isomorphisme de Scott)
Soit $\mathcal{I}$ une interprétation pour $\mathcal{L}_I$. Il existe un énoncé ϕ de $\mathcal{L}_{w_1w}$ telle que pour toutes les interprétations dénombrables de $\mathcal{I}^*$ de $\mathcal{L}_I$, $\mathcal{I}^* = \phi$ si et seulement si $\mathcal{I}^*$ est isomorphe à $\mathcal{I}$.

Intuitivement, ce théorème exprime qu'un énoncé infinitaire peut parfaitement caractériser une interprétation dénombrable de $\mathcal{L}_I$. La preuve standard implique entre autres la construction de conjonctions infiniment longues - à l'extérieur de $\mathcal{L}_{w_1w}$ - de formules atomiques exprimant une vérité à propos des éléments du domaine de $\mathcal{L}_I$. Par exemple, si le domaine de $\mathcal{I}$ est $\mathbb{N}$ et que $\mathcal{I}$ possède la relation $>$ - ordre strict - alors $(3, 2), (4, 3),\ldots$ sont des éléments de $>$. Il doit par conséquent exister une formule atomique correspondant

à chacun de ces éléments si l'on veut exprimer $\mathcal{I}$, et la conjonction de ces formules - qui n'est qu'une partie de la preuve du théorème - devient avec la relation G interprétée comme $>$ et les symboles c_i interprétés comme des constantes

$$G_{c_3c_2} \wedge G_{c_4c_3} \wedge \ldots$$

ou, avec la notation des conjonctions infiniment longues propre à $\mathcal{L}_{w_1w}$

$$\bigwedge\{G_{c_ic_j}/c_i, c_j \text{ sont des constantes } \& \ \mathcal{I} = G_{c_ic_j}\}$$

Pour Bringsjord, ce raisonnement mathématique requiert que l'on raisonne *avec* un *langage de la pensée* qui est similaire à $\mathcal{L}_{w_1w}$. D'après lui, l'objection selon laquelle on pourrait uniquement raisonner *à propos de* $\mathcal{L}_{w_1w}$ et non *avec* $\mathcal{L}_{w_1w}$ est donc fausse.

Contrairement à la première objection, la seconde objection adressée à Bringsjord est d'après moi très convaincante car elle souligne la distinction entre raisonner *à propos de* et raisonner *avec* qui est au cœur du problème soulevé par la possibilité pour l'être humain d'hyper-calculer. A l'inverse de Bringsjord, je pense en effet que l'on ne peut pas raisonner *avec* $\mathcal{L}_{w_1w}$ et cela même s'il peut être possible de raisonner *à propos de* $\mathcal{L}_{w_1w}$.

Dans le but de défendre mon affirmation, je vais formuler deux objections contre la thèse des hyper-cerveaux à partir de la distinction entre raisonner *à propos de* et raisonner *avec*. Bien que ces deux objections possèdent globalement la même structure que l'objection présentée ci-dessus au sens où elles ont pour but de montrer qu'il est impossible de raisonner *avec* $\mathcal{L}_{w_1w}$, elles ne sont pas fondées sur la notion de langage mais sur les notions de système déductif et de machine théorique.

Objection 2'. La première de ces deux objections est fondée sur la notion de système déductif. Commençons par un ensemble de définitions :

Définition (Raisonner *à propos de* ou *avec* un système déductif) . *Raisonner à propos d'un système déductif signifie déduire des théorèmes de ce système en utilisant des preuves vérifiables ou non vérifiables.*

▪ *Raisonner avec un système déductif signifie déduire des théorèmes de ce système en utilisant uniquement des preuves vérifiables.*

Définition (Preuve et règles vérifiables)

▪ *Une preuve est vérifiable lorsqu'elle ne comporte que des règles vérifiables.*

▪ *Une règle vérifiable est une règle dont les étapes élémentaires peuvent être vérifiées une à une par un être humain.*

Exemple 1 (Règles vérifiables et non vérifiables)

▪ Un exemple de règle vérifiable est la règle de l'arithmétique de Peano qui affirme que si n est un entier naturel alors son successeur est aussi un entier naturel : $N(n) \Longrightarrow N(s(n))$ où N est la propriété d'être un entier naturel et $\Longrightarrow$ l'inférence logique. Chacune des étapes élémentaires de cette règle est vérifiable puisque la fonction successeur $s(n)$ est primitive récursive, c'est-à-dire qu'il existe une procédure effective qui permet au calculateur de vérifier chaque étape du calcul de $s(n)$.

▪ Un exemple de règle non vérifiable est l'ω-règle de l'ω-logique [25]. L'ω-logique est un système déductif qui comporte tous les axiomes et les règles de la logique du premier ordre avec en plus la règle infinitaire suivante : pour toute formule $P(x)$ où x est une variable libre, $P(0), P(1), P(2), \ldots \Longrightarrow \forall x\ (N(x) \rightarrow P(x))$. L'$\omega$-règle n'est pas vérifiable car un être humain ne peut pas vérifier une à une l'infinité d'étapes élémentaires de cette règle.

Ces définitions permettent de formuler précisément l'objection **2'** : il est possible de raisonner *à propos de* $\mathcal{L}_{w_1 w}$ car l'on peut déduire des théorèmes comme le théorème de Scott à partir de règles non vérifiables telle que celles permettant de construire des conjonctions ou des disjonctions infinies. En revanche, il n'est pas possible de raisonner *avec* $\mathcal{L}_{w_1 w}$ puisque ce système peut contenir des preuves infiniment longues qui ne sont pas vérifiables [Barwise, 1969], [Ebbinghaus et al., 1984]. Passons maintenant à la seconde objection.

Objection 2". Cette objection est construite à partir de la notion de machine théorique au moyen d'une traduction des concepts utilisés dans l'objection **2'**. Pour ce faire, je propose la traduction et les définitions suivantes :

25. A ne pas confondre avec l'Ω-logique [Woodin, 1999].

Système	Déduire	Preuve	Théorème	Règle
↓	↓	↓	↓	↓
Machine	Calculer	Calcul	Fonction calculable	Etape de calcul

Définition (Raisonner *à propos de* ou *avec* une machine)

▪ *Raisonner à propos d'une machine théorique signifie calculer des fonctions calculables par la machine en utilisant des calculs vérifiables ou non vérifiables.*

▪ *Raisonner avec une machine signifie calculer des fonctions calculables par la machine en utilisant uniquement des calculs vérifiables.*

Définition (Calcul et étape de calcul vérifiables)

▪ *Un calcul est vérifiable lorsqu'il ne comporte que des étapes de calcul vérifiables.*

▪ *Une étape de calcul est vérifiable si chacune de ses étapes élémentaires peut être vérifiée une par une par un être humain.*

Exemple 2 (Etapes de calcul vérifiables et non vérifiables)

▪ Un exemple trivial d'étape de calcul vérifiable est le mouvement vers la droite de la tête de lecture d'une MT. Cette étape est naturellement vérifiable par un être humain car (1) elle est par définition une étape de calcul élémentaire ; et (2) c'est une des opérations que peut exécuter une MT - qui est une formalisation de la notion de procédure effective.

▪ Un exemple d'étape de calcul non vérifiable peut être donné à partir de la MTI (Machine à Temps Infinis) pouvant exécuter en une étape ω étapes de calcul [Hamkins and Lewis, 2000]. Puisqu'une telle machine peut exécuter une étape contenant ω étapes élémentaires, il est impossible pour un être humain de vérifier une par une l'infinité d'étapes élémentaires que contient une telle étape.

Voici à présent l'objection **2"** formulée *via* la traduction et les définitions précédentes : il est possible de raisonner *à propos d'*une MTI car l'on peut calculer des fonctions calculables par la machine en utilisant des calculs vérifiables. En effet, puisque par définition la MTI est une extension de la MT, elle est capable de calculer les fonctions Turing-calculables à l'aide de calculs

vérifiables. Par exemple, la fonction $f : \mathbb{N}^2 \to \mathbb{N}$ définie par $f(x, y) = x \sqcup y$ où $\sqcup$ est l'opération de concaténation - par exemple $23 \sqcup 11 = 2311$ - ne demande qu'un nombre fini d'étapes élémentaires pour être calculée. En revanche, il n'est pas possible de raisonner *avec* une MTI puisque cette machine peut exécuter des calculs non vérifiables qui demandent à l'être humain d'exécuter un nombre infini d'étapes élémentaires - notamment le calcul de la fonction arrêt propres aux MT qui non Turing-calculable.

Le but des objections **2'** et **2"** est double. Elles ont d'une part pour but d'insister sur le fait qu'un être humain ne peut pas raisonner *avec* des éléments tels que les étapes d'une preuve ou les étapes d'un calcul qui seraient infinis. En particulier, un être humain ne peut pas raisonner avec $\mathcal{L}_{w_1 w}$ puisque ce système permet de construire des preuves non vérifiables. Leur but est d'autre part de montrer que la question de savoir si un être humain est capable de raisonner avec $\mathcal{L}_{w_1 w}$ repose sur la possibilité pour un être humain d'exécuter une infinité d'étapes de calcul, capacité qui rendrait les preuves de $\mathcal{L}_{w_1 w}$ vérifiables. Autrement dit, tant qu'une preuve montrant qu'un être humain est capable d'exécuter une infinité d'étapes ne sera pas apportée, les objections **2'** et **2"** énoncées à l'encontre de l'argument de Bringsjord resteront pertinentes.

Toutefois, le premier argument venant à l'esprit et qui va à l'encontre d'une telle preuve consiste à dire que le cerveau où sont effectués les calculs est un mécanisme fini à la fois en espace - il existe un nombre fini de neurones et de liaisons synaptiques - et en temps - la vie d'un être humain est limitée. En d'autres termes, comment un être humain pourrait exécuter une infinité d'étapes de calculs s'il ne dispose que de ressources - temps et espaces - qui sont finies ?

Il est aisé de comprendre que ce n'est pas aux mathématiques mais à la physique d'évaluer cette dernière question. C'est pourquoi les deux prochains arguments que j'étudie sont fondés sur la physique afin de montrer que le cerveau est capable d'hyper-calcul.

2.4 Processus physiques du cerveau et simulation effective

Contrairement à l'argument de Bringsjord qui se fonde sur le raisonnement mathématique, Penrose défend la thèse des hyper-cerveaux à partir d'arguments physiques [Penrose, 1989]. Voici une présentation de ces arguments suivie d'une explication des principaux problèmes qu'ils soulèvent.

2.4.1 L'argument de Penrose

L'idée principale de Penrose est de dire qu'il existe des processus physiques exécutés par le cerveau qui ne peuvent pas être simulés par une MT. Schématiquement, un processus physique n'est pas simulable par MT (non Turing-simulable) si le comportement entrée-sortie de ce processus ne peut pas être reproduit par une MT. Plus précisément, l'argumentation de Penrose est construite de la façon suivante :

1. Certains cristaux particuliers appelés *quasi-cristaux* ne sont pas Turing-simulables car une de leurs caractéristiques, leur croissance, est impossible à décrire à partir du comportement entrée-sortie d'une MT.
2. Il existe une forte similitude entre d'une part la croissance des quasi-cristaux et d'autre part la croissance et la contraction des épines dentritiques du cerveau.
3. Par conséquent, la croissance et la contraction des épines dentritiques du cerveau ne sont pas Turing-simulables.

Je vais tenter d'expliquer, à partir d'arguments informels, les principales étapes du raisonnement de Penrose. En premier lieu, qu'est-ce qu'un quasi-cristal ? Un quasi-cristal est un solide qui possède un spectre de diffraction essentiellement discret mais dont la structure n'est pas périodique. Les cristaux, définis encore récemment comme des structures périodiques - c'est-à-dire telles que les positions des atomes se répètent de la même manière dans tout le matériau - ont des figures de diffraction discrètes, et il a longtemps été implicite qu'un matériau dont la figure de diffraction est discrète est nécessairement périodique. Plus précisément, ceci équivaut à dire que sa

figure de diffraction ne peut admettre que des symétries d'ordre 2, 3, 4 ou 6, propriété qui est connue sous le terme de *restriction cristallographique*. Cependant, de nombreuses symétries *interdites* - c'est-à-dire incompatibles avec une structure périodique - ont été observées par la suite, condamnant définitivement la restriction cristallographique [Shechtman et al., 1984]. Enfin le terme de quasi-cristal s'est imposé progressivement pour désigner ces matériaux non périodiques qui diffractent comme des cristaux.

Maintenant, pourquoi Penrose pense-t-il que la croissance des quasi-cristaux n'est pas simulable par une MT ? Pour ce dernier, la non Turing-simulabilité des quasi-cristaux proviendrait de la propriété suivante : les quasi-cristaux croîtraient de manière *non locale*, c'est-à-dire que

> La croissance d'un quasi-cristal [d'après le modèle de Penrose] requiert que les atomes situés dans des cellules très éloignées interagissent d'une certaine manière afin de communiquer leurs positions et leurs orientations relatives [Stephens and Goldman, 1991, p. 47].

La non localité attribuée aux quasi-cristaux faisant actuellement l'objet de toutes les controverses, mon but ne sera pas d'apporter une réponse à la question de leur mode de croissance. Je vais néanmoins expliquer pourquoi la non localité est une propriété en faveur de la non Turing-simulabilité.

Il existe en effet un lien explicite entre non Turing-calculabilité, non Turing-simulabilité et non localité. En particulier, la non localité est une condition suffisante à la non Turing-simulabilité et donc à la non Turing-calculabilité. Ce lien provient plus précisément du travail de Gandy issu de son article *Church's Thesis and Principles for Mechanisms* [Gandy, 1980]. Dans cet article, Gandy essaie de fournir un argument en faveur de la thèse suivante :

Thèse M Tout ce qui peut être calculé par une machine est calculable [par une machine de Turing] [Gandy, 1980, p. 124].

L'auteur clarifie immédiatement son énoncé en considérant uniquement *les dispositifs mécaniques discrets et déterministes* (DMDD), qui sont « de manière schématique des ordinateurs digitaux » [Gandy, 1980, p. 126] :

Thèse de Gandy Toutes les fonctions qui peuvent être calculées par un DMDD sont Turing-calculables.

La première étape de l'argument de Gandy est de formuler la notion de DMDD en termes de contraintes ou *principes* selon la terminologie de l'auteur. J'expose ci-dessous ces principes en mettant de côté les aspects techniques propres à la présentation de Gandy :

1. **Principe de description** : tout DMDD M peut être décrit par le couple $< S, F >$, où S est une classe structurelle - un ensemble potentiellement infini d'états - et F est une opération de transformation de S_i vers S_j. Ainsi, si S_0 est l'état initial de M alors $F(S_0), F(F(S_0))$ ... sont ses états suivants.
2. **Principe de la hiérarchie limitée** : la structure S est hiérarchisée en un nombre fini de niveaux.
3. **Principe du réassemblage unique** : la structure S est décomposable en un nombre fini de parties qui ont chacune une taille finie et qui ne contiennent qu'un nombre fini de sous-parties.
4. **Principe de localité** : la structure hiérarchique de la machine admet une description topologique telle que l'état d'une partie de la machine $F^n(S)$ lors de son évolution ne dépend que de l'état des parties appartenant à son voisinage dans $F^{n-1}(S)$.

La seconde étape de Gandy est de prouver le théorème selon lequel toute fonction calculable par un dispositif qui satisfait les 4 principes est Turing-calculable. Les étapes qui constituent la preuve de Gandy peuvent être résumées de la façon suivante [26] :

1. **Thèse P** : un DMDD satisfait les 4 principes.
2. **Théorème** : les fonctions calculables par un dispositif satisfaisant les 4 principes sont Turing-calculables.
3. **Thèse de Gandy** : les fonctions calculables par un DMDD sont Turing-calculables.

26. Il n'est pas nécessaire de rentrer dans les détails de la preuve de Gandy pour remplir mes objectifs. Le lecteur peut tout de même consulter [Sieg and Byrnes, 1999] et [Sieg, 2002] pour une version détaillée mais simplifiée de la preuve de Gandy.

Revenons maintenant au principe de localité. Ce principe est justifié par Gandy de la façon suivante :

> Dans l'analyse de Turing, la condition selon laquelle l'action est déterminée uniquement à partir d'une portion finie de mémoire a été fondée sur une limitation humaine. Nous replaçons cette condition par une limitation physique que nous appellerons *principe de causalité locale*. Sa justification repose sur la vitesse finie de propagation des effets et des signaux : la physique contemporaine rejète la possibilité d'une action instantanée à distance [Gandy, 1980, p. 135].

En fait, même si le principe de localité est énoncé par Gandy dans un cadre mathématique, ce principe est une abstraction de deux hypothèses physiques,

> qu'il existe une borne inférieure sur les dimensions linaires de chaque partie atomique du dispositif et qu'il existe une borne supérieure (la vitesse de la lumière) sur la vitesse de propagation des échanges [Gandy, 1980, p. 126].

En effet, si la propagation de l'information admet une borne supérieure alors un atome doit transmettre ou recevoir de l'information en un temps fini provenant de son voisinage dont les limites sont finies. De la même manière, s'il existe une borne inférieure sur la taille des atomes, le nombre d'atomes à proximité d'un atome est fini.

Ces deux hypothèses physiques sont d'autant plus justifiées qu'elles sont des conditions nécessaires à la Turing-calculabilité. Dit autrement, une machine physiquement construite qui ne satisferait pas le principe de localité serait capable de calculer des fonctions non Turing-calculables [Copeland and Shagrir, 2007]. Afin d'illustrer ce fait, présentons une hyper-machine (HM) qui ne satisfait pas le principe de localité : l'HM de Davies [27].

Davies considère un univers qui obéit aux lois de Newton et dans lequel la matière peut être divisée à volonté [Davies, 2001]. Son intention

27. La présentation de l'HM de Davies est plus complexe et élégante que celle que je m'apprête à faire. Cette HM sera toutefois étudiée plus en détails au chapitre suivant.

est de construire une série de machines dont les opérations physiques sont « compatibles avec la physique étudiée dans les années 1850 » [Davies, 2001, 672]. D'une part, même si la vitesse de propagation du signal admet une borne supérieure, la construction de la machine ne dépend pas de cette vitesse qui peut être arbitrairement élévée. D'autre part, cette construction ne dépend ni d'une quantité infinie de matériel disponible ni d'une quantité infinie d'énergie pouvant être stockée dans un volume donné. Elle repose en revanche sur l'absence d'une borne inférieure concernant la taille des composants atomiques. En effet, l'astuce de la construction réside dans l'idée que chaque nouvelle machine est une version miniaturisée de la machine précédente. Ainsi, la condition émise par Gandy vis-à-vis du principe de localité selon laquelle il existe une borne inférieure sur les dimensions linaires de chaque partie atomique du dispositif n'est pas satisfaite.

Dans le détail, Davies propose une procédure afin de construire une série de machines $M_1, M_2, M_3 \ldots$ où chaque machine est une machine mécanique - au sens de Babbage c'est-à-dire constituée de rouages, câbles et autres composants mécaniques - travaillant avec un robot pouvant produire à la fois une nouvelle version de la machine et une nouvelle version de lui-même. La nouvelle machine M_{n+1} qui inclue aussi un nouveau robot est de taille plus petite que la machine M_n permettant ainsi à M_{n+1} d'être construite plus rapidement que M_n. Plus généralement, la taille des machines suit une série géométrique décroissante de la même manière que le temps nécessaire à leur construction, ce qui rend la taille de l'ensemble des machines à peine supérieure de celle de la première machine.

En supposant que chacune des machines ainsi construites est équivalente à une MT universelle, montrons comment une série infinie de telles machines peut calculer la fonction arrêt propre aux MT. Etant donné un argument (m, n), les machines commencent par simuler les opérations de la m-ième MT sur la donnée en entrée n. Chaque machine exécute une seule opération, M_1 exécute la première, M_2 la seconde, et ainsi de suite. Après avoir exécutée l'opération, chaque machine vérifie si la MT qui est simulée atteint son état d'arrêt. Dans le cas où la MT atteint son état d'arrêt, cette dernière envoie un signal à M_1 ; dans le cas contraire, M_1 produit une nouvelle machine qui

exécute la prochaine opération. En supposant que la MT qui est simulée ne s'arrête jamais, la simulation prendra la forme d'une série infinie de simulations exécutées en un temps finie. Par conséquent, M_1 *attendra* au plus un temps fini avant de recevoir une réponse - signal ou absence de signal - et pourra imprimer sur son ruban une réponse - arrêt ou non - en fonction de la MT simulée.

En résumé, le principe de localité causale est une condition nécessaire à la Turing-calculabilité puisqu'en supprimant ce principe, il est possible de concevoir des machines telles que celle proposée par Davies qui sont capables de calculer des fonctions non Turing-calculables [28].

L'analyse de Gandy conforte ainsi la première partie de l'argumentation de Penrose selon laquelle la non localité de la croissance des quasi-cristaux rendrait cette croissance non Turing-simulable. Mais qu'en est-il de la seconde partie ? L'argument de la seconde partie, je le rappelle, est de dire qu'il existe une forte similitude entre la croissance des quasi-cristaux et la croissance /contraction des épines dentritiques du cerveau. C'est sur ce point que la cohérence de l'argumentation a subi le plus de critiques car les arguments en faveur de cette similitude sont fondés sur des spéculations :

> Je vais prendre des risques et spéculer que cette contraction [des dentrites] ou cette croissance pourrait être gouvernée par quelque chose de semblable aux processus impliqués dans la croissance des quasi-cristaux [Penrose, 1989, pp. 566-567].

Ces spéculations sont plus précisément fondées sur une analogie qui est plausible selon Penrose. Cette analogie consiste à dire que la croissance d'un quasi-cristal est fortement influencée par les concentrations d'atomes et d'ions qui sont à proximité du point de croissance. De manière similaire, Penrose envisage que la croissance ou la contraction des familles d'épines dentritiques seraient elle aussi influencée par les substances neurotransmettrices variées qui sont émises autours de ces épines dentritiques par les neurones. Voici à présent certains problèmes que soulèvent ces spéculations.

28. Consulter [Copeland and Shagrir, 2007] pour plus de détails.

2.4.2 Objections

La position de Penrose soulève deux importants problèmes qui découlent des trois conditions devant être satisfaites afin de montrer qu'il existe des processus exécutés par le cerveau qui ne sont pas simulables par MT. Ces trois conditions sont (1) que l'on construise une preuve mathématique montrant que la croissance des quasi-cristaux n'est pas Turing-simulable ; (2) que l'on exhibe des éléments pertinents en faveur de la similitude entre la croissance des quasi-cristaux et celle des épines dentritiques ; et (3) que l'on montre que cette similitude est assez forte pour impliquer la non Turing-simulabilité de la croissance des épines dentritiques.

Plus précisément, le premier problème soulevé par la position de Penrose concerne le caractère informel voire schématique des arguments qu'il propose. Par exemple, aucune des trois conditions énoncées ci-dessus n'a été démontrée par l'auteur et chacune de ces conditions repose sur des arguments informels [Penrose, 1994]. Le second problème est plus grave encore car il concerne la possiblité de satisfaire les trois conditions, possibilité qui ne fait pas l'unanimité parmi les scientifiques.

A propos du caractère non local de la croissance des quasi-cristaux, des chercheurs de Carnegie Mellon ont en effet proposé un modèle théorique de ces cristaux fondé sur des processus aléatoires qui a le même pouvoir prédictif que le modèle de Penrose mais qui ne requiert que des règles de croissances locales [Widom et al., 1987]. En outre, le débat est encore plus vivace au sujet de l'hypothèse selon laquelle la croissance des épines dentritiques est similaire à celle des quasi-cristaux et donc qu'elle implique la non Turing-simulabilité. Penrose fonde en particulier son argumentation sur l'idée que les processus non Turing-simulables qui ont lieu au sein du cerveau proviendraient d'une nouvelle physique appelée *gravité quantique* qui unifierait à la fois la mécanique quantique et la relativité générale :

> Ce que je suis en train d'affirmer concerne la physique inconnue qui gouverne les limites de ces théories [quantique et relativité générale], à savoir la théorie non encore découverte de la gravité quantique [Penrose, 1989, p. 568].

Le problème sous-jacent à la gravité quantique est qu'aucune des théories proposées dans le but de la décrire ne fait l'unanimité parmi les physiciens [Rovelli, 2004], [Penrose, 2005]. Plus précisément, les principales théories faisant l'objet d'importantes recherches sont au nombre de quatre :

1. La supergravité [West, 1986].
2. La théorie des supercordes [Green et al., 1987].
3. La gravité quantique à boucles [Rovelli, 1998].
4. La théorie des twisteurs [Penrose, 1967].

La grande diversité de ces différentes approches est donc un frein à l'argument de Penrose selon lequel la non Turing-simulabilité des processus du cerveau provient des propriétés de la gravité quantique. En particulier, cet argument ne pourra pas être prouvé tant que l'on aura pas réussi à unifier la mécanique quantique et la relativité générale en une théorie physique fondamentale, unification qui n'est pas à l'ordre du jour.

Il existe cependant une dernière tentative en faveur de la thèse des hypercerveaux qui permet de contourner le problème de l'unification de la mécanique quantique et de la relativité générale. Cette tentative menée par Siegelmann n'est pas axée sur la physique mais sur la biologie car elle a pour but de montrer que l'organisation biologique du cerveau humain peut être modélisée par une HM.

2.5 Réseaux de neurones et calcul analogique

Le modèle proposé par Siegelmann est fondé sur les qui sont inspirés de leurs contreparties biologiques. On doit toutefois être prudent et souligner qu'il existe au sein des travaux de Sielgelmann une tension au sujet de la nature de son modèle. Il n'est pas aisé en effet de déterminer si son modèle à été proposé pour des raisons purement mathématiques ou bien s'il a pour vocation de modéliser à proprement parler le cerveau humain. Néanmoins, puisque

> Les réseaux de neurones artificiels ont été inspirés via leurs contreparties biologiques, il est naturel de s'interroger sur les conséquen-

ces que pourraient avoir ces réseaux sur le biologique [Zénil and Hernandez-Quiroz, 2006, p.2].

2.5.1 L'argument de Siegelmann

Siegelmann introduit son modèle fondé sur les réseaux neuronaux dans son livre intitulé *Neural Networks and Analog Computation : Beyond the Turing Limit* [Siegelmann, 1999, p. 19]. Un réseau neuronal est considéré comme un ensemble fini de N *processeurs* - ou neurones artificiels. Ces processeurs sont élémentaires au sens où ils ne peuvent pas être davantage décomposés en terme d'opérations [29]. Chaque processeur possède un état $x_i(t)$ au temps t où t indice un temps discret. Si deux processeurs sont connectés entre eux, alors ces derniers forment un *nœud* possédant une valeur appelée *poids* qui représente une certaine quantité - par exemple une résistance ou un potentiel. On suppose qu'au temps $t = 0$ tous les poids sont initialisés, que les processeurs - ou nœuds - sont connectés d'une certaine façon et qu'une donnée en entrée est fournie au réseau. Le réseau est alors représenté à chaque instant t par un vecteur u_j de dimension M dont les composants sont des éléments de $\{0, 1\}$. En détails, le comportement du réseau peut être décrit à chaque étape de la façon suivante :

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}x_j(t) + \sum_{j=1}^{M} b_{ij}u_j(t) + c_i\right)$$

Les a_{ij}, b_{ij} et c_i sont les poids du réseau possédant chacun une valeur initiale fixée par le concepteur du réseau. Ces valeurs initiales sont essentielles pour l'argument de Siegelmann car la puissance calculatoire du réseau - c'est-à-dire sa capacité à calculer plus ou mons de fonctions - dépend du type de poids utilisé - entiers naturels, rationnels ou réels. Dans ce qui suit, je vais tout particulièrement m'intéresser aux cas pour lesquels les poids sont des

29. Pour saisir cette notion de décomposition, prenons un exemple simple. En théorie de la calculabilité, une opération non décomposable est la fonction successeur $s(x)$ avec $x \in \mathbb{N}$. Inversement, l'opération somme est décomposable car elle peut être définie en termes d'opérations élémentaires telle que la fonction successeur. La fonction somme $+(x, y)$ où $x, y \in \mathbb{N}$ peut être en effet définie par $+(x, 0) = x$ et $+(x, s(y)) = s(+(x, y))$.

nombres réels conformément à l'argumentation de Sielgelmann.

La fonction σ de la précédente équation est la *fonction d'activation*. Les fonctions d'activation sont très importantes pour le fonctionnement du réseau. En voici une qui est longtemps discutée par Siegelmann dans son livre - voir pp. 20-21 pour davantage de fonctions d'activation.

$$\sigma(x) = \begin{cases} 0 \text{ si } x < 0, \\ x \text{ si } 0 \leq x \leq 1, \\ 1 \text{ si } x > 1. \end{cases}$$

Après avoir sélectionné une fonction d'activation, on termine par choisir un nombre l de processeurs que l'on considère comme les données en sortie du réseau.

Siegelmann élabore dans son ouvrage un protocole qui permet à ces réseaux de calculer des fonctions et d'accepter des langages. Pour ce faire, il est nécessaire de diviser les données en entrée en deux ensembles : l'un est nommé *donnée* noté D et l'autre *validation* noté V. D'une part, l'ensemble D véhicule un signal binaire au sein du réseau. Ce signal est codé par 1 s'il est élévé, et est codé par 0 s'il est faible. D'autre part, l'ensemble V indique que l'ensemble D est actif, c'est-à-dire véhicule un signal élevé si la donnée en entrée est présente et véhicule un signal faible autrement. La donnée en entrée du réseau est ainsi codée par un signal à partir de D et V et peut représenter toutes formes d'informations - nombres, symboles... Par exemple si l'on souhaite insérer en entrée le nombre 42 en base 10, V doit tout d'abord véhiculer un signal élevé à $t = 0$. Puis de $t = 1$ à $t = 5$ les *bits* 1,0,1,0,1,0 sont envoyés par D. Enfin à $t = 6$, V véhicule un signal faible pour indiquer que le codage de la donnée en entrée est terminé. De la même manière, il existe deux ensembles G et H analogues à D et V qui sont utilisés pour les données en sortie [30].

Je peux maintenant passer à un certain nombre de définitions. Je commence par définir les notions de mots et de langages acceptés par un réseau - notions fréquemment utilisées par Siegelmann - puis je défini la notion de

30. S'il n'y a pas d'ambiguité, j'écrirai à partir de maintenant que *D est élévé / faible* à la place de *D véhicule un signal élévé / faible*.

fonction calculable par un réseau.

Définition (Mot référencé/accepté/rejeté par un réseau)
Etant donnés des poids appropriés, un mot est référencé par le réseau - à $t = r$ - si

1. *Ce mot est codé à l'état initial par les ensembles D et V.*
2. *H est élevé pour $t = r$ et faible pour tout $t < r$.*

De plus, si G est élevé pour $t = r$ alors le mot est accepté, autrement le mot est rejeté.

Définition (Langage accepté/rejeté par un réseau)
Un langage est accepté par un réseau si tous les mots du langage sont acceptés par le réseau et que les mots appartenant au complément de ce langage sont rejétés. Le langage est autrement rejeté.

Définition (Fonction calculable par un réseau)
Une fonction - partielle - $\varphi(x)$ est calculable par un réseau si pour tout argument x présenté comme donnée en entrée au réseau :

1. *Lorsque $\varphi(x)$ n'est pas définie, H reste faible - les données en sortie valent 0. G peut en revanche fluctuer entre ces deux états mais puisque H n'est jamais élevé, l'état de G n'a pas de conséquence sur la donnée en sortie du réseau.*
2. *Lorque $\varphi(x)$ est définie, il existe un nombre r - le temps de réponse - tel que :*
   (a) *G fournit en sortie les symboles successifs de $\varphi(x)$ entre les temps r et $r+|x|-1$, où $|x|$ est la longueur de la suite de symboles codant x.*
   (b) *H reste élevé entre les temps r et $r + |x| - 1$, et reste faible pour tous les autres temps.*

Je viens de présenter de façon générale la manière dont un réseau de Siegelmann calcule une fonction. je vais à présent expliquer comment certains de ces réseaux sont capables d'hyper-calculer.

Siegelmann introduit un système de trois réseaux calculant à partir de câbles et de portes logiques sur l'ensemble $\{0, 1\}$. Ces trois réseaux servent à simuler des familles de circuits - qui sont des ensembles de portes logiques - et sont définis de la façon suivante :

1. Le *réseau d'entrée* prend la donnée en entrée appropriée dans les circuits simulés et le mémorise.
2. Le *réseau d'extraction* prend une donnée en entrée appropriée à partir du réseau d'entrée et le simule sur le circuit approprié.
3. Le *réseau de sortie* coordonne les deux premier réseaux et fournie la donnée en sortie finale.

C'est le réseau d'extraction qui est le plus discuté par Siegelmann car ce réseau est la seule partie qui exécute des calculs allant au-delà de la MT [31]. Plus précisément, la puissance du réseau dépend du type de poids utilisés par le réseau d'extraction. Siegelmann insiste en effet sur le fait que pour que le réseau d'extraction marche, il est suffisant que tous les poids du réseau sauf un - que j'appelle C - soient décrits par des nombres rationnels plutôt que par des nombres réels arbitraires. Toutefois, puisque C peut être un nombre réel arbitraire il suffit que C code un nombre non Turing-calculable de portes logiques afin d'être lui-même un nombre non Turing-calculable. Il est ainsi possible de décrire de quelle manière le réseau peut hyper-calculer : si on interprète la valeur de C comme un nombre réel non Turing-calculable, tous les langages sont acceptés dans des circuits de tailles exponentielles, où la taille est définie par le nombre total de portes logiques [Siegelmann, 1999, p. 16]. Notons cependant que chaque famille de circuits doit être simulé par une valeur différente de C et donc par des réseaux différents. Il suit que chaque langage accepté requiert un réseau différent. Plus généralement, voici un tableau récapitulant la puissance de calcul des réseaux de Siegelmann en fonction des types de poids utilisés [Zénil and Hernandez-Quiroz, 2006] :

31. Je ne vais pas entrer dans les détails de la construction du réseau d'extraction mais le lecteur peut se référer à [Siegelmann, 1999, p. 65] pour plus de détails

Types de poids	Puissance de calcul
Nombres réels non Turing-calculables	Au-delà de la MT
Nombres rationnels	Au plus équivalent à la MT
Nombres entiers relatifs	Au plus inférieure à la MT

2.5.2 Objections

Le modèle de Siegelmann fait l'objet de nombreuses critiques mais deux d'entre elles sont particulièrement pertinentes. Tandis que la première critique soulève un problème qui ne remet pas en cause la thèse de Siegelmann proprement dite, la seconde critique apporte quant à elle des doutes qui affaiblissent drastiquement les résultats apportés par Siegelmann.

Première problème. Le premier problème qui doit être soulevé concerne le fait que Siegelmann ne donne aucune preuve de son résultat selon lequel si C est interprété comme un nombre réel non Turing-calculable alors tous les langages sont acceptés dans des circuits de tailles exponentielles. Dans le but de soutenir son résultat, l'auteur fait en effet appel à l'ouvrage de Balcàzar, Dìaz et Cabarrò [Balcàzar et al., 1995]. Cependant, cet ouvrage ne soutient pas le résultat de Siegelmann de manière explicite.

La raison en est que l'ouvrage de Balcàzar, Dìaz et Cabarrò (B-D-C) traite principalement de la complexité algorithmique et non de la calculabilité effective. En particulier, ces derniers utilisent des HM - et plus précisément des OM - dans le seul but d'augmenter la vitesse des calculs et non afin de calculer des fonctions non Turing-calculables. Siegelmann au contraire interprète leur résultat - l'existence de bornes de complexité pour la simulation de certaines familles de circuits - pour des fonctions qui ne sont pas calculables par MT. Le problème est que B-D-C ne discutent à aucun moment des capacités hyper-calculatoires de leurs circuits et l'introduction de leur livre est très clair à ce sujet : leurs travaux ne concernent pas la théorie de la calculabilité. Ainsi, les travaux de B-D-C ne permettent pas de soutenir la thèse selon laquelle les réseaux de Siegelmann peuvent recconnaître tous les langages possibles sur $\{0,1\}^*$.

Siegelmann fournit néanmoins dans son livre une véritable preuve qui concerne l'équivalence de ses réseaux avec un certain type de MT nommées (). Une ATM est similaire à une machine de Turing à oracle car elle dispose d'une suite de symbole w_n - *the advice* - qui dépend uniquement de la longueur de la donnée en entrée n et qui peut être utilisée lors d'un calcul comme un oracle [Karp and Lipton, 1980]. Une ATM polynomiale - où la longueur de w_n est polynomiale en n - peut par exemple calculer les fonctions appartenant à la classe de complexité **P/poly** qui contient des fonctions non Turing-calculables comme la fonction arrêt pour les fonctions unaires. Plus encore, une ATM exponentielle - où la longueur de w_n est exponentielle en n - peut calculer n'importe quelles fonctions $f : \{0,1\}^* \to \{0,1\}$.

Sielgelmann démontre en particulier que ses réseaux sont équivalents aux ATM polynomiaux. Même si ce résultat est suffisant pour montrer que ses réseaux hyper-calculent, il reste néanmoins plus faible que son affirmation précédente selon laquelle ses réseaux sont équivalents aux ATM exponentielles.

Second problème. Le second problème concernant le modèle de Siegelmann est moins technique que le précédent mais est paradoxalement plus percutant. Ce problème a été originellement introduit par Davis :

> Et le seul moyen qui permet aux réseaux de Siegelmann d'espérer reconnaître un langage non calculable est d'utiliser des poids non calculables. Les réseaux de neurones peuvent aller 'au-delà de la limite de Turing' si on leur fournit au préalable des poids qui sont déjà non calculables ! [Davis, 2004, p. 203].

Ici Davis met en lumière deux failles propres à ces réseaux :

1. Les poids non calculables ont une origine inconnue. En effet, quelle est l'origine des poids non calculables ? Puisque Siegelmann ne clarifie à aucun moment d'où proviennent les poids initiaux permettant à ses réseaux d'hyper-calculer, il semble donc que son argument soit circulaire.
2. Le résultat de Siegelmann n'est à proprement parler pas surprenant. En

> effet, les liens entre la calculabilité et les différentes classes de nombres ont déjà été analysés par Turing tout d'abord en 1936 puis en 1939 [Turing, 1936], [Turing, 1939]. Turing a par exemple montré en 1936 que les nombres irrationnels utilisés en mathématiques tels que e et π sont Turing-calculables. Mais c'est dans sa thèse de 1939 que Turing élabore ce que l'on nomme aujourd'hui *la théorie des degrés d'indécidabilité* qui permet d'analyser la puissance d'une MT en fonction des types de nombres - poids - qu'elle utilise pour calculer. En particulier, on peut montrer à partir de cette théorie qu'une MT utilisant des nombres non Turing-calculables peut calculer plus qu'une MT standard. Par exemple, si ce nombre code les résultats de la fonction arrêt propre aux MT alors la machine utilisant ce nombre sera capable de décider si une MT arbitraire travaillant sur une donnée particulière s'arrête ou non. Par conséquent, le résultat selon lequel un réseau calcule du calculable à l'aide de poids Turing-calculables et du non calculable à l'aide de poids non Turing-calculables n'est en lui-même pas une avancée mathématique.

La première faille mise en lumière par Davis, à savoir l'origine inconnue des poids non Turing-calculables, est un sérieux obstacle à la thèse selon laquelle les réseaux de Siegelmann décrivent de manière adéquate les limites du cerveau humain. Même si cette faille ne remet pas en cause selon moi le modèle de Siegelmann en tant qu'HM, elle insiste néanmoins sur le fait que rien ne permet de conclure que les réseaux modélisent correctement le cerveau humain. En plus de ne pas donner d'arguments convaincants en faveur des capacités hyper-calculatoires du cerveau humain, les réseaux sont soumis à la seconde faille qui illustre de son côté que les résultats de Siegelmann n'apportent rien de plus que ce qui est déjà connu :

> Devons-nous conclure que Siegelmann est une pionnière de l'hyper-calcul ? En fait, comme nous pourrons le voir, il y a bien moins de choses à dire à propos de sa position qu'il n'y parait à première vue [Davis, 2004, p. 202].

En conclusion, même si les réseaux de Siegelmann peuvent être considérés comme calculant plus que la MT, ils ne permettent en aucun cas de conclure que le cerveau humain peut être modélisé par ces réseaux [32].

Avoir réfuté les principaux arguments en faveur de la thèse des hyper-cerveaux - à savoir les arguments de Lucas, Bringsjord, Penrose et Siegelmann - est selon moi suffisant pour conjecturer que l'être humain n'est pas capable d'hyper-calculer de lui-même. Il existe bien sûr d'autres arguments qui tentent de montrer que l'esprit humain est supérieur à la MT [33], mais la capacité de ces derniers à apporter des preuves en faveur de la thèse des hyper-cerveaux est marginale comparée à celle des arguments que je viens de réfuter. Par conséquent, je supposerai dans la suite de ce travail que le cerveau humain n'est pas capable d'hyper-calcul.

A partir de cette hypothèse, je peux à présent déduire le résultat suivant : l'être humain ne peut pas avoir un accès aux valeurs de fonctions non Turing-calculables, et cela même en principe. Il est ainsi nécessaire d'utiliser un dispositif externe au cerveau humain pour accéder à ces valeurs. En d'autres termes, l'être humain ne peut pas se limiter aux formalisations d'HM s'il souhaite hyper-calculer car seule une HM physiquement construite peut fournir les valeurs de fonctions non Turing-calculables.

Ce résultat atteste de la nécessité de prendre en compte les dispositifs capables d'hyper-calcul afin de définir les limites de ce qui est calculable. Tandis que les ordinateurs actuels n'influencent pas les limites définies par les logiciens des années 1930, la construction d'HM pourraient au contraire les modifier. Si tel était le cas, cela aurait pour conséquence de replacer le concept général d'ordinateur au centre de l'étude des limites du calcul, que l'on considère ces limites en pratique ou en principe.

32. Pour davantage d'arguments en faveur de cette conclusion, voir [Douglas, 2003].
33. Voir notamment [Scriven, 1953], [McCall, 1999], [Bringsjord and Arkoudas, 2004] et [Redhead, 2004].

3 Conclusion

Dans ce chapitre, j'ai défendu la thèse selon laquelle la construction d'une HM est nécessaire pour hyper-calculer, c'est-à-dire pour avoir accès aux valeurs de fonctions non Turing-calculables. J'ai plus précisément soutenu cette thèse dans le but de montrer que les limites du calculable ne sont pas indépendantes de la construction des ordinateurs contrairement à ce qui est soutenu depuis la seconde moitié du XX^{e} siècle.

L'analyse des logiciens des années 1930 a en effet permis de montrer que même si les ordinateurs sont en pratique indispensables, ils n'ont en principe aucune influence sur les limites du calcul. Pour comprendre ce point fondamental, j'ai commencé par définir précisément ce qu'est un ordinateur car la nécessité d'utiliser un ordinateur pour calculer dépend de ce que l'on considère comme étant un ordinateur. J'ai ensuite expliquer (1) que les ordinateurs sont en pratique nécessaires pour calculer ; et (2) qu'ils sont néanmoins contingents lorsque l'on considère les calculs qui peuvent être exécutés en principe par un être humain.

Plus précisément, la puissance de calcul d'un ordinateur le rend en pratique indispensable pour effectuer des calculs qui demandent trop de ressources physiques. Toutefois, cette puissance n'est en principe plus nécessaire. D'une part, les ordinateurs sont des dispositifs universels au sens où les calculs qu'ils peuvent effectués sont exactement ceux pouvant être exécutés par une MT. D'autre part, les travaux de Turing ont permis de montrer que les calculs pouvant être effectués par une MT peuvent être exécutés en principe par un être humain. L'ordinateur n'est donc pas nécessaire en principe puisqu'un être humain peut exécuter tous les calculs d'un ordinateur.

Contrairement à la construction des ordinateurs, j'ai ensuite défendu la thèse selon laquelle la construction des HM est en pratique et en principe indispensable pour hyper-calculer. Mon principal argument a été d'affirmer que l'être humain n'est en principe pas capable d'hyper-calculer, et donc qu'il ne peut pas faire abstraction aux ordinateurs comme dans le cas du calcul effectif. Premièrement, j'ai expliqué que mon argument reposait sur la vérité de la *thèse des hyper-cerveaux*, thèse selon laquelle le cerveau humain

peut hyper-calculer. Deuxièmement, j'ai tenté de défaire cette thèse en montrant que les principaux arguments en sa faveur n'étaient pas capables de la démontrer.

La conclusion de ce chapitre, à savoir que la construction d'une HM est nécessaire pour hyper-calculer, permet selon moi de réfuter la thèse selon laquelle la construction d'ordinateurs n'apporterait rien de nouveau concernant les limites en principe du calcul. Etudier les ordinateurs devient ainsi indispensable pour établir les limites du calculable, que ces limites soient définies en principe ou en pratique. En ce sens, la distinction en principe / en pratique sur laquelle se fondent les logiciens des années 1930 n'est pas pertinente pour établir des limites indépendantes des avancées technologiques.

Cependant, ma position philosophique sur les ordinateurs ne peut être validée *que* si une HM est physiquement construite. En effet, il se pourrait que la construction physique d'une HM soit physiquement impossible. Cette impossibilité pourrait par exemple provenir d'une incompatibilité entre les processus exécutés par les HM et les théories physiques ou bien d'une insuffisance technologique que l'on ne pourrait jamais combler. Si tel était le cas, les limites érigées par Turing et Church resteraient à la fois intactes et indépendantes de la construction des ordinateurs. Par conséquent l'hypercalcul repose *in fine* sur la construction d'une HM.

Quelles sont les tentatives qui ont été proposées dans le but de construire une HM ? Sur quelles théories physiques se fondent ces tentatives ? En existe-t-il une plus prometteuse que les autres ? Ces questions font l'objet du prochain chapitre.

Chapitre 3

Démontrer la thèse physique de l'hyper-calcul

> How can we ever exclude the possibility of our being presented, some day (perhaps by some extraterrestrial visitors), with a (perhaps extremely complex) device or 'oracle' that 'computes' a noncomputable function ?
>
> Martin Davis

Le précédent chapitre a permis de montrer que la construction physique d'une hyper-machine (HM) est une condition nécessaire pour franchir la barrière de Turing. Plus précisément, c'est l'accès aux valeurs de fonctions non Turing-calculables qui dépend d'une telle construction : puisque l'être humain n'est en principe pas capable d'hyper-calculer, l'accès à ces valeurs doit nécessairement se faire par l'intermédiaire d'un dispositif pouvant hyper-calculer, à savoir d'une HM.

La thèse selon laquelle il est possible de construire une HM sera nommée la () :

Thèse (Thèse physique de l'hyper-calcul)
Il est possible de construire physiquement une HM.

Mais que signifie exactement *construire physiquement une HM* ? Voici comment je définis ce qu'est une HM physiquement construite :

Définition (HM physiquement construite)
Une HM physiquement construite est un dispositif qui satisfait les conditions suivantes :

1. *Le dispositif a pour formalisation mathématique une HM - par exemple une des HM présentées au chapitre 1.*
2. *Le dispositif est physiquement possible au sens où son fonctionnement est compatible avec les lois énoncées par une théorie physique.*
3. *Le dispositif est physiquement constructible au sens où l'on doit disposer des ressources physiques nécessaires à sa construction [Vaidya, 2007].*
4. *Une fois construit le dispositif est capable de calculer au moins une fonction non Turing-calculable.*

La TPHC est cruciale pour l'hyper-calcul car si cette thèse s'avère être fausse, alors la possibilité d'hyper-calculer se dissoudrait avec elle. Pour être plus précis, c'est la possibilité *pour un être humain* d'hyper-calculer qui serait condamnée : s'il est impossible de construire un dispositif capable d'hyper-calcul alors les valeurs de fonctions non Turing-calculables lui seront à jamais inaccessibles -puisque l'être humain ne peut pas hyper-calculer de lui-même ; si cette construction est en revanche possible alors les fonctions qu'il peut calculer ne se limiteraient pas aux fonctions Turing-calculables.

L'objectif de ce chapitre n'est pas de déterminer si la TPHC est vraie ou fausse - objectif qui dépasse le cadre de cette thèse - mais de répondre à la question suivante : est-ce que les propositions actuelles de construction d'HM pourraient être un jour réalisées afin de démontrer la TPHC ?

Une réponse positive à cette question aurait des conséquences à la fois philosohiques et pratiques. Une réponse positive permettrait en effet de valider ma position philosophique selon laquelle l'étude des ordinateurs - ici des dispositifs capables d'hyper-calcul - est indispensable pour établir les limites

du calculable, que ces limites soient définies en principe ou en pratique. Une HM physiquement construite permettrait ainsi à l'être humain d'effectuer - pour la première fois de son histoire - des calculs situés au-delà de la barrière de Turing.

Une réponse positive ne serait pas moins dénuée d'intérêts pratiques. Prenons l'exemple concret de la cryptographie qui est une discipline s'attachant à protéger des messages en assurant confidentialité, authenticité et intégrité [1]. Si la construction d'un dispositif capable d'hyper-calcul est possible, alors il est aisé d'imaginer quelles pourraient être ses applications en cryptographie. Un message que l'on souhaite protéger pourrait par exemple être codé à partir des valeurs d'une fonction non Turing-calculable. Ce codage garantirait une confidentialité presque parfaite puisque ni l'être humain ni un ordinateur - par opposition à une HM - ne seraient en principe capables de décrypter le message, et cela quels que soient leurs puissances de calcul.

Il est cependant honnête de dire que la communauté scientifique est très critique vis-à-vis de la démonstration de la TPHC. Ses adversaires soutiennent en effet que la construction d'une HM soulève autant de problèmes liés à sa compatibilité avec les théories physiques que de problèmes liés à la technologie nécessaire à sa construction. A titre d'exemple, les scientifiques s'accordent à dire que les HM issuent de la physique newtonienne ne pourront jamais être construites car les propriétés physiques sur lesquelles elles sont fondées n'existent pas dans le monde réel :

> C'est comme si le mot 'impossible' était vu comme un *challenge*. Bien que les lois de la thermodynamique ont démontré que la recherche d'une machine en mouvement perpétuel est un exercice futile, des inventeurs affirmant avoir construit de tels dispositifs assiègent encore les offices des brevets et réussissent à obtenir des soutiens financiers provenant d'investisseurs crédules [Davis, 2004, p. 195].

1. Schématiquement, la cryptographie offre trois types de garantie. *L'authenticité* : le destinataire d'un message signé est assuré que son émetteur est bien celui qu'il prétend être. *La confidentialité* : l'émetteur du message chiffré est assuré que son destinataire sera seul à pouvoir le lire. *L'intégrité* : le contenu du message n'a subi aucune altération entre son envoi et sa réception [Delfs and Knebl, 2007].

Les scientifiques apportent ainsi davantage de crédit à une réponse négative à la question d'une possible démonstration de la TPHC - cette réponse négative aurait néanmoins un intérêt pratique, celui d'éviter à l'investissement alloué à la recherche de se perdre dans un projet irréalisable.

Pourquoi consacrer ce chapitre à l'évaluation des propositions de construction d'une HM alors que la communauté scientifique semble être très sceptique quant à leur réalisation ? Parce qu'une de ces propositions est selon moi particulièrement prometteuse : l'HM fondée sur l'utilisation de l'aléatoire [Stannett, 2003], [Calude, 2005].

Cette proposition est d'après moi prometteuse car à l'inverse des propositions rivales, les difficultés soulevées par cette HM sont davantage d'ordre conceptuel que physique. En effet, les principales tentatives de construction en vue de démontrer la TPHC ont pour inconvénient d'impliquer de sérieux problèmes physiques. Par exemple, l'accumulation d'une quantité infinie de données qui est nécessaire pour construire certaines HM est hautement controversée [Earman and Norton, 1993, p. 40]. D'un point de vue théorique en revanche, ces tentatives expliquent généralement correctement de quelles manières elles peuvent hyper-calculer.

La situation est néanmoins inversée dans le cas de l'HM fondée sur l'aléatoire car c'est la théorie et non pas la pratique qui pose problème. Même s'il est actuellement possible de produire de l'aléatoire - dans un sens précis qui sera défini - l'utilisation de l'aléatoire comme source d'hyper-calcul soulève des problèmes théoriques qui affaiblissent la thèse selon laquelle l'HM peut hyper-calculer. Pour être plus précis, ces problèmes théoriques sont liés au rôle de l'aléatoire au sein de l'hyper-calcul et plus généralement au sein du calcul. C'est en particulier parce que ces liens ne sont pas clairs qu'il subsiste actuellement des doutes quant à la capacité pour cette proposition d'hyper-calculer. Ainsi, comme le constate Calude

> Il est surprenant que nous disposions d'une méthode qui marche dans le monde physique, mais qui est soumise à des difficultés qui concernent sa justification d'un point de vue mathématique [Calude, 2005, p.12].

Bien que l'HM fondée sur l'utilisation de l'aléatoire soit confrontée à des

obstacles théoriques, je pense néanmoins que ces obstacles peuvent être surmontés dans leur grande majorité. Je pense en outre que cette proposition est plus prometteuse que les autres.

Le présent chapitre aura pour objectif de défendre ces deux dernières affirmations. Plus particulièrement, il est divisé en trois sections. Dans les deux premières sections, je présente et discute les propositions concurrentes à l'HM fondée sur l'aléatoire. Ces propositions correspondent aux HM conçues à partir des théories newtonienne, relativiste et quantique. Même si ces deux premières sections sont très techniques, elles sont néanmoins nécessaires pour comprendre les problèmes théoriques et pratiques que soulèvent ces propositions. Je consacre enfin la dernière section à l'HM fondée sur l'aléatoire qui est d'après moi plus prometteuse que ses rivales newtonienne, relativiste et quantique.

1 Propositions newtoniennes et relativistes

Les propositions conçues à partir des théories newtonienne et relativiste sont en général toutes fondées sur l'exécution d'un *supertask*[2] (ST). Il existe toutefois une différence majeure entre les propositions fondées en ce qui concerne leurs chances respectives d'être un jour physiquement construites. Contrairement aux HM relativistes, la probabilité de pouvoir un jour construire une HM newtonienne est en effet quasiment nulle. La raison en est que les HM newtoniennes utilisent des propriétés propres à la physique newtonienne qui ne régissent pas le monde réel. Par exemple, ces HM se fondent sur l'hypothèse newtonienne selon laquelle l'espace peut être divisé à volonté ce qui rentre en contradiction avec la structure atomique de la matière du monde réel. Du côté des HM relativistes, la situation est néanmoins différente au sens où les espace-temps qui fournissent les propriétés nécessaires à la conception de ces HM sont physiquement possibles même si personne ne possède de preuve de leur existence. Les propositions newtoniennes et relativistes partagent en revanche un point commun, celui d'être soumis à des limites

2. Pour rappel, un ST est l'exécution d'une infinité d'étapes de calcul en un temps fini - *cf.* chapitre 1.

technologiques qui n'ont pas été franchies actuellement.

1.1 Hyper-calcul newtonien

Comme je viens de l'écrire, les propositions conçues à partir de la physique newtonienne sont généralement toutes fondées sur l'exécution d'un ST [Earman and Norton, 1996]. La raison pour laquelle les HM newtoniennes sont fondées sur le principe des ST réside dans l'hypothèse de continuité inscrite au sein de cette théorie. D'après cette hypothèse, l'espace et le temps sont continus au sens où ils peuvent être divisés à volonté en conservant leurs propriétés.

Pour avoir une première idée du fonctionnement d'un ST en physique newtonienne, voici une description très simple introduite par Laraudogoitia [Laraudogoitia, 1996],[Laraudogoitia, 1998]. Ce dernier décrit un système composé d'un ensemble infini de sphères élastiques possédant chacune une masse m. Ces sphères sont positionnées le long d'un intervalle ouvert *via* le principe suivant : la sphère σ_i est positionnée au point $x_i = 1/2^i$, où $i = 1, 2, 3, \ldots$ Au point $x_0 = 1$ se trouve une autre sphère - notée σ_0 - ayant la même masse m et qui se déplace vers σ_1 avec une vitesse constante v. Après un temps fini, les deux sphères σ_0 et σ_1 entrent en collision : σ_0 s'arrête au point $x_1 = 1/2$ tandis que σ_1 commence à se déplacer vers σ_2 avec une vitesse constante v. σ_1 et σ_2 entrent ensuite en collision et échangent leurs énergies, c'est-à-dire que σ_1 perd son énergie cinétique et s'arrête à $x_2 = 1/4$, tandis que σ_2 commence à se déplacer. L'enchaînement de collisions entre les sphères continu ainsi tout au long de l'intervalle. A la fin de l'expérience, toutes les sphères σ_i seront arrêtées et chaque sphère i sera positionnée au point $1/2^{i+1}$.

Même si Laraudogoitia décrit clairement une expérience qui conduit à l'exécution d'un ST, il n'explique toutefois pas comment cette expérience permet de calculer des fonctions non Turing-calculables. Pour comprendre comment concevoir une authentique HM en physique newtonienne, je détaille ci-dessous la principale proposition énoncée à l'intérieur de cette théorie, à savoir l'HM de Davies [Davies, 2001].

1.1.1 La proposition de Davies

L' est l'exemple type des propositions newtoniennes car elle possède les principales caractéristiques inhérentes à ces propositions. L'HM de Davies est tout d'abord cohérente d'un point de vue théorique au sens où elle est (1) compatible avec les lois de la physique newtonienne ; et (2) capable de calculer des fonctions non Turing-calculables. Cependant, elle implique de sérieux problèmes physiques qui remettent sévèrement en cause sa possible construction. Afin d'illustrer plus précisément ces deux caractéristiques, voici une présentation de l'HM de Davies.

Davies propose de construire une machine M_1 qui possède quatre éléments :

1. Un ordinateur n'ayant que des parties mécaniques. Davies introduit ce premier élément pour une raison pratique : « Afin d'éviter toute considération relative à l'électricité ou au magnétisme nous supposons que la machine est mécanique » [Davies, 2001, p. 672].

2. Un chronomètre c_1.

3. Un espace mémoire de m_1 bits.

4. Un robot pouvant produire à la fois un nouvel ordinateur et un clone de lui-même.

Le dernier élément - à savoir le robot - peut en particulier produire une nouvelle machine M_2. Cependant, la nouvelle machine n'est pas totalement identique à l'ancienne. M_2 possède en effet une mémoire de taille $m_2 = 2m_1$ et ses composants sont supposés être 16 fois plus petits que ceux de M_1. Les composants de M_2 ont donc une taille $s_2 = s_1/8$. Enfin, le temps c_2 qui est inscrit sur le chronomètre de M_2 est $c_2 = c_1/8$.

Puisque M_2 est identique à M_1 au niveau de son fonctionnement, M_2 peut elle aussi construire une nouvelle machine M_3. Si ce processus continue cela donne ainsi naissance à une hiérarchie infinie de machines ayant les paramètres suivants : $c_{n+1} = c_n/8$, $m_{n+1} = 2m_n$ et $s_{n+1} = s_n/8$. On peut par exemple déduire à partir de ces paramètres que la taille de l'ensemble des

machines est au plus de

$$s_1 \sum_{n=0}^{\infty} 8^{-n} = \frac{8s_1}{7} < \infty$$

Mais il est aussi possible de calculer le temps que met chaque machine à être manufacturée. Soit t_n le temps nécessaire pour construire M_n. D'une part, puisque les composants de M_{n+1} sont 16 fois plus petits que ceux de M_n et que les outils à l'intérieur de M_n se déplacent à la même vitesse que ceux à l'intérieur de M_{n-1}, les composants doivent être créés à chaque fois 16 fois plus rapidement. D'autre part, l'espace mémoire de M_{n+1} est deux fois plus important que celui de M_n. Par conséquent, le temps complet nécessaire pour construire M_{n+1} est 8 fois moins important que le temps nécessaire pour construire M_n, à savoir $t_{n+1} = t_n/8$.

Passons maintenant à la manière dont l'HM de Davies peut calculer des fonctions non Turing-calculables. Schématiquement, l'HM exécute un ST pour parcourir en un temps fini l'ensemble infini des solutions possibles à un problème. Davies prend comme exemple le dernier théorème de Fermat qui établit qu'il n'existe pas de 4-tuples (a, b, c, x) avec $x > 2$ et $a, b, c \neq 0$ qui satisfait l'équation $a^x + b^x = c^x$. L'idée est de montrer comment l'HM peut tester l'ensemble infini des propositions possibles et ainsi démontrer par une recherche exhaustive que le théorème de Fermat est vrai. En d'autres termes, si l'HM trouve une solution vérifiant l'équation pour un 4-uples le théorème de Fermat est faux.

Dans les détails, le calcul de l'HM implique une suite de propositions $\mathcal{P}_n$ qui nécessite de plus en plus de calculs au fur et à mesure que la longueur de chaque proposition augmente - (a, b, c, x) sont testés par ordre croissant. Plus précisément, le problème général est fourni à la machine M_1 et la procédure suivante est appliquée. Tout d'abord M_1 tente de résoudre le problème - trouver une solution à l'équation - pour $n = 1$, et inscrit Y si elle trouve effectivement une solution. Dans le cas contraire, M_1 construit M_2 et lui transmet le problème entier ainsi que le nombre $n = 2$. M_2 se comporte ensuite de la même manière que M_1. Autrement dit, la machine M_n tente de résoudre à l'étape n le problème pour le cas n. Enfin d'après la convergence

de la série géométrique qui représente le temps de calcul de l'HM, une des M_n saura après un intervalle de temps fini que le problème n'a pas de solution. M_n informera alors M_{n-1} qu'il n'existe pas de solution et cette dernière machine transmettra le message tout au long de la chaîne de machines jusqu'à M_1 qui inscrira N.

La stratégie qui vient d'être présentée peut bien sûr être appliquée pour résoudre le problème de l'arrêt propre au MT. Il suffit pour cela de remplacer l'équation de départ par une MT particulière et les propositions $\mathcal{P}_n$ par les données en entrée que peut prendre la MT. Après un intervalle de temps fini, l'HM de Davies saura si la MT s'arrête ou non pour une donnée en entrée n particulière - consulter [Davies, 2001, p. 676] pour davantage de détails.

Même si le fonctionnement de cette HM est tout à fait correct d'un point de vue théorique, elle est cependant très critiquée en pratique. Plus précisément, elle implique de sérieux problèmes qui compromettent sa possible réalisation physique. Loin de présenter l'ensemble de ces problèmes, je vais me concentrer sur certains points illustrant clairement les difficultés liées à la construction physique de l'HM de Davies.

1.1.2 Problèmes physiques

Je vais présenter trois problèmes physiques soulevés par l'HM de Davies. Le premier problème pourrait être surmonté en théorie tandis que le second n'est pour l'instant pas résolu. Le dernier problème réduit quant à lui pratiquement à zéro la probabilité de voir se réaliser la proposition de Davies.

Stockage des données. Le premier problème qui est soulevé par l'HM concerne la quantité de mémoire dont dispose chaque machine M_n. Pour comprendre d'où provient le problème, reprenons la résolution du dernier théorème de Fermat présentée ci-dessus.

Au cours de cette résolution, on remarque que la quantité de mémoire que doit posséder chaque machine M_n augmente. En effet, puisque le nombre de propositions $\mathcal{P}_n$ augmente à chaque étape de calcul, la quantité de symboles devant être mémorisée augmente elle aussi.

Le stockage de ces symboles n'est toutefois pas un problème en soi puisque l'augmentation de la quantité de symboles est inférieure à la quantité de mémoire dont dispose chaque machine. Plus précisément, la différence entre le nombre de symboles des propositions $\mathcal{P}_{n+1}$ et $\mathcal{P}_n$ est inférieure à la différence entre la quantité de mémoire des machines M_{n+1} et M_n. La raison en est que la mémoire d'une nouvelle machine doublera - $m_n = 2m_{n-1}$ - contrairement au nombre de symboles d'une nouvelle proposition.

En revanche, le nombre d'étapes de calcul augmente lui aussi au cours du temps et il se pourrait que les données liées au calcul augmentent plus rapidement que la quantité de mémoire des machines M_n. Cette hypothèse est justifiée car Davies présuppose « que la longueur des calculs n'augmente pas rapidement en fonction de n » [Davies, 2001, p. 674]. L'auteur explique en particulier que « [. . .] le calcul croît en fonction de n à une vitesse qui n'est pas contrôlable » (p. 675) ; il semblerait donc que la mémoire dont dispose les machines soit insuffisante pour exécuter le calcul.

Davies réussit tout de même à surmonter ce problème lié à la mémoire grâce à l'astuce suivante. Il suffit d'insérer dans le programme des machines l'instruction selon laquelle si la proposition $\mathcal{P}_n$ est testée par la machine M_r, alors M_r s'arrête après 2^r étapes de calculs même dans le cas où elle n'a pas terminée de tester $\mathcal{P}_n$. Si M_r n'a pas terminé de tester la proposition, elle construit alors la machine M_{r+1} et lui transmet le problème à résoudre *sans* augmenter la valeur de n. En d'autres termes, une machine M_r construit M_{r+1} si une des conditions suivantes est satisfaite : (1) elle n'a pas trouvé de solution satisfaisant $\mathcal{P}_n$; (2) sa mémoire est pleine ; ou (3) elle a effectué 2^r étapes de calculs. De cette façon, la machine ne sera jamais à court d'espace mémoire.

Transmission des données. Le second problème est lié à la transmission des données entre les machines. En particulier, l'HM nécessite une transmission des données entre les machines M_n qui soit dénuée d'erreurs malgré l'augmentation de la vitesse de transmission tout au long de la hiérarchie de machines. Pour permettre à la transmission de s'opérer sans erreurs, Davies suppose qu'aucune quantité de bruit ne vient perturber cette transmission.

Plus précisément, Davies suppose que les machines M_n n'ont aucun contact avec l'extérieur de sorte que le système - la hiérarchie de machines - puisse être considéré comme un système fermé, à savoir comme un système pour lequel il n'y pas d'échange - donc de perte - d'énergie avec l'extérieur.

Cette dernière supposition est toutefois très audacieuse car les données fournies à la première machine de la hiérarchie doivent provenir de l'extérieur. En effet, même si un seul bit d'information est nécessaire à la première machine pour commencer son calcul, ce bit provient nécessairement de l'extérieur ce qui empêche de fait de considérer l'HM comme un système fermé. De son côté, Davies tente de proposer une solution à ce problème mais qui n'est pas tout à fait convaincante :

> [...] le premier bit d'information pourrait être envoyé par une machine à oracle. Une telle machine vivrait dans un univers hautement idéalisé où elle pourrait par exemple fonctionner à une température égale à zéro [Davies, 2001, p. 678].

Puisque l'existence d'une telle machine à oracle n'est pas à prendre en compte sérieusement, le problème du bruit lié à la transmission des données entre les machines M_n n'est actuellement pas résolu.

Miniaturisation des composants. Le dernier problème auquel fait face l'HM de Davies est de loin le plus décisif. Pour reprendre les mots de son concepteur : « Le problème essentiel est de savoir si une machine peut produire une réplique d'elle-même » [Davies, 2001, p. 673].

Le problème n'est cependant pas exactement celui de savoir si une machine est capable de produire une réplique d'elle-même. Précisément, la proposition de Davies suppose plus que la capacité de produire une réplique d'une machine : elle suppose (1) de produire une réplique *plus petite* que l'originale ; et (2) de continuer cette miniaturisation *ad infinitum*. C'est en particulier la satisfaction de la seconde condition qui est au cœur du problème.

Continuer la miniaturisation *ad infinitum* suppose en effet que l'espace de l'univers réel soit divisible à volonté. Or cet univers est atomique ce qui implique l'existence de limites physiques sur la taille des particules de matière. Plus précisément, l'univers réel n'est pas newtonien mais quantique à l'échelle

microscopique. La physique quantique permet en particulier d'établir que la miniaturisation d'un des composants de la machine de Davies - son chronomètre - admet une limite - appelée *limite inférieure de Wigner* - à partir de laquelle on peut déduire une taille et un poids minimum que doit posséder un chronomètre pour fonctionner [Barrow, 1996]. Cette limite empêche ainsi l'HM de Davies d'être construite puisqu'il est impossible de miniaturiser à volonté les chronomètres des machines M_n sans altérer leur fonctionnement.

En résumé, la proposition de Davies ne pourra jamais voir le jour à cause des propriétés physiques fondamentales qui régissent l'univers réel. D'une manière générale, on peut dire que les propositions fondées sur la physique newtonienne sont vouées à l'échec à cause de la théorie newtonienne elle-même. Puisque cette dernière ne correspond pas au monde réel, toute construction physique de machines fondées sur la théorie newtonienne est par conséquent impossible à réaliser.

La cause de l'impossibilité de construire physiquement les HM issues de la physique newtonienne, à savoir l'incompatibilité de cette théorie avec le monde réel, ne s'applique pas aux HM qui ont été conçues au sein des physiques relativiste et quantique. Toutefois, ces propositions sont soumises à de nombreuses autres critiques. Voici une analyse des propositions relativistes.

1.2 Hyper-calcul relativiste

Les propositions relativistes sont toutes fondées sur des espace-temps particuliers appelés *espace-temps de Malament-Hogarth* qui pourraient être utilisés dans le but d'exécuter un ST et ainsi résoudre des problèmes indécidables. De telles propositions soulèvent néanmoins de nombreuses critiques qui ne concernent pas directement la théorie de la relativité proprement dite mais la possibilité physique des espace-temps de Malament-Hogarth et des HM conçues pour les utiliser.

Afin de présenter les propositions relativistes et les critiques dont elles font l'objet, je procède en deux mouvements. Dans un premier mouvement, je commence par expliquer pourquoi la majorité ces propositions se fondent sur l'utilisation des espace-temps de Malament-Hogarth pour hyper-calculer.

J'expose ensuite dans un second moment la principale tentative de construction d'une HM qui utilise ces espaces-temps : les *machines à trous noirs*.

1.2.1 Les espaces temps de Malament-Hogarth

La possibilité physique des ST au sein de la théorie de la relativité générale est analysée par Earman [Earman, 1995]. Même si ses travaux ne concernent pas directement l'hyper-calcul au sens où il n'utilise pas les ST pour calculer des fonctions non Turing-calculables, Earman présente néanmoins plusieurs espace-temps capables de fournir un cadre théorique à l'exécution d'un nombre infini d'étapes en un temps fini. Schématiquement, un ST peut être exécuté à l'aide de deux machines opérant dans ces espace-temps. La première machine dispose d'une quantité infinie de temps tandis que la seconde ne peut en utiliser qu'une quantité finie. Plus précisément, le *cône de lumière* du passée de la seconde machine contient la *ligne d'univers* de la première machine [3]. La première machine peut ainsi effectuer une infinité d'étapes calculatoires et ensuite communiquer le résultat de son calcul à la seconde machine.

En fait, c'est Pitowsky qui a en premier élaboré une expérience de pensée afin de démontrer que l'on peut aller au-delà de la MT à l'aide de la théorie de la relativité générale [Pitowsky, 1990]. Supposons qu'il n'existe aucune restriction liée à la taille de l'espace de calcul. Supposons de plus qu'Angelene soit obnubilée par le problème suivant qui fut énoncé par Wittgenstein et dont la réponse ne peut être fournie par une MT : est-ce que l'expansion décimale de π comporte trois 7 consécutifs ? Dans le but de répondre à cette question, Angelene voyage vers un satellite qui parcourt son orbite autour de la Terre. Le satellite en question possède un moteur d'une telle puissance que sa vitesse

3. En théorie de la relativité, *l'espace-temps* est une collection de points appelés *évènements* qui possèdent quatre coordonnées : une coordonnée de temps et trois coordonnées d'espace. Le *cône de lumière* d'un évènement est un cône double qui créé la distinction entre son passé et son futur. Soit E un évènement, le cône de lumière du futur de E est composé de toutes les trajectoires de lumière qui débutent de E et qui voyagent dans le futur, tandis que le cône de lumière du passé de E est composé de toutes les trajectoires de lumière qui s'arrête à E et qui proviennent du passé. Enfin, *la ligne d'univers* d'un objet est une suite d'évènements dans l'espace-temps correspondant à l'historique de l'objet.

tangentielle instantanée est égale à $v(t) = c\sqrt{1 - e^{-2t}}$, où t est le temps dans le référenciel terrien et c la vitesse de la lumière. D'une part, si τ désigne le temps local du satellite, on a $d\tau = e^{-t}dt$. D'autre part, puisque $\int_0^\infty e^{-t}dt = 1$ on conclue que lorsqu'une seconde s'écoule sur le satellite, une quantité infinie de temps s'écoule sur Terre. Voici à présent la stratégie mise en place pour résoudre le problème de Wittgenstein. Pendant qu'Angelene parcourt son orbite autour de la Terre sur le satellite, ses étudiants examinent l'expansion décimale de π pour déterminer si elle contient trois 7 consécutifs. Lorsque ces personnes deviennent trop âgées pour poursuivre le travail - c'est-à-dire dans le cas où elles n'ont toujours pas trouvé trois 7 consécutifs - elles demandent à leurs jeunes étudiants de continuer à examiner l'expansion décimale de π. Ce processus se poursuit jusqu'à l'un des deux dénouements suivants. Si trois 7 consécutifs sont découverts alors la nouvelle est immédiatement transmise au satellite ce qui laissera à Angelene une seule seconde pour retourner sur Terre. En revanche, si aucun message ne parvient à Angelene au bout d'une seconde - c'est-à-dire au bout d'une quantité infinie de temps écoulée sur Terre - elle saura qu'il n'existe pas trois 7 consécutifs au sein de l'expansion décimale de π. Cependant, à la fin de ce second dénouement Angelene sera détruite sous l'effet des forces gravitationnelles.

L'expérience de Pitowsky est analysée en détails par Earman [Earman, 1995]. Même si ce dernier définit formellement les espace-temps qui permettrait à la proposition de Pitowsky d'aboutir - appelés *espace-temps de Pitowsky* - Earman soulève néanmoins deux problèmes liés à l'expérience. Le premier problème a pour origine le fait qu'Angelene est soumise lors de l'expérience à une magnitude de l'accélération qui est égale à $e^t/\sqrt{1 - e^{-2t}}$. Selon Earman, cela signifie qu'Angelene sera probablement écrasée par de gigantesques forces gravitationnelles après un court intervalle de temps. En d'autres termes, il est fort possible qu'Angelene ne sache jamais s'il existe trois 7 consécutifs au sein de l'expansion décimale de π, et cela même si une telle suite existe. Le second problème est plus important car il concerne la procédure de calcul permettant de résoudre le problème de Wittgenstein et donc de calculer du non Turing-calculable.

La procédure proposée par Pitowsky possède en effet un problème lié au

fait qu'Angelene ne reçoit aucun message dans le cas où l'expansion décimale de π ne contient pas trois 7 consécutifs. Plus précisément, si la réponse au problème de Wittgenstein est affirmative alors Angelene reçoit un message anonçant que trois 7 consécutifs ont été trouvés ; autrement elle ne reçoit aucun message. Cependant, la signification de l'absence de message n'est pas claire du point de vue d'Angelene. En particulier, puisqu'elle ne reçoit aucun message signifiant qu'il n'existe pas trois 7 consécutifs, Angelene ne peut pas être sûre que la réponse au problème de Wittgenstein est négative. En d'autres termes, même si l'expérience permet de résoudre le problème de Wittgenstein aucun observateur ne pourra affirmer que la réponse à ce problème est négative.

Pour contourner ce second problème, la solution est de proposer une expérience capable d'enlever toute ambiguité concernant l'absence de réponse. Cela peut être fait en remplaçant les espace-temps de Pitowsky par d'autres espace-temps particuliers appelés [Hogarth, 1992]. Ces espace-temps ont la particularité de faire abstraction de l'observateur humain - d'Angelene - et donc de rendre non ambigue l'absence de réponse. Expliquons à présent en détails l'expérience qui permet d'hyper-calculer à partir des espace-temps de Malament-Hogarth.

La conception de cette expérience est due à Shagrir & Pitowsky et elle permet de décrire une HM pouvant calculer la fonction arrêt des MT [4] [Shagrir and Pitowsky, 2003, p. 88]. Plus précisément, l' est constituée de deux ordinateurs modernes T_A et T_B pouvant communiquer entre eux. Grâce aux propriétés des espaces-temps de Malament-Hogarth, au moment où T_A aura effectué un nombre fini d'étapes calculatoires, T_B aura quand à lui calculé un nombre infini d'étapes. Maintenant, voici la procédure permettant à l'HM de calculer la fonction arrêt h des MT :

1. On commence par fournir la donnée en entrée n à T_A qui la transmet à T_B. T_B est une MT universelle, c'est-à-dire que son programme lui permet de simuler le calcul de la $n^{\text{ème}}$ MT opérant sur n.

4. Dans ce qui suit, la fonction arrêt h est définie de la façon suivante. Soient $M_1, M_2, \ldots, M_m$ une énumération des MT, la fonction h est maintenant définie par $h(n) = 1$ si la $n^{\text{ème}}$ MT s'arrête sur n et $h(n) = 0$ autrement.

2. Lors du calcul, si T_B s'arrête à un moment donné, ce dernier envoie immédiatement un signal à T_A. Si T_B ne s'arrête jamais, il n'envoie jamais de signal à T_A.
3. Enfin, lorsque que T_B aura effectué un nombre infini d'étapes en un temps fini, T_A inscrira 1 s'il a reçu un signal de T_B et insicrira 0 autrement. Par conséquent, l'HM a la capacité de calculer la fonction non Turing-calculable h définie par $h(n) = 1$ si la $n^{\text{ème}}$ MT s'arrête sur n et $h(n) = 0$ autrement.

L'HM proposée par Shagrir et Pitowsky possède bien la propriété voulue : l'observateur utilisant l'HM recevra un signal - 0 ou 1 - quelle que soit la réponse - affirmative ou non - du problème indécidable à résoudre. Autrement dit, si l'on remplace le problème de l'arrêt des MT par le problème de Wittgenstein, l'observateur saura à coup sûr si l'expansion de π contient ou non trois 7 consécutifs.

Même si l'HM qui vient d'être décrite surmonte le problème soulevé par Earman, sa possible construction demande néanmoins de pouvoir mémoriser une quantité infinie de données dans l'espace mémoire de l'HM. Il existe actuellement deux propositions permettant de fournir une quantité infinie de données. La première proposition consiste à compresser cette quantité infinie de données dans un espace fini [Moore, 1990]. La seconde option est quant à elle de mémoriser cette quantité infinie de données dans un espace infini. En théorie, certains espace-temps tels que les *espace-temps de Reissner-Nordström* qui sont des espace-temps de Malament-Hogarth ne seraient pas limités dans l'espace. En pratique cependant la construction d'un dispositif pouvant supporter une quantité infinie de mémoire est impossible à moins de violer certaines lois physiques fondamentales [Earman and Norton, 1993, p. 40].

Malgré les problèmes physiques liés à la possible construction de l'HM introduite par Shagrir et Pitowsky, d'autres tentatives ont vu le jour. Ces dernières, bien que différentes de celle qui vient d'être présentée, sont elles aussi fondées sur l'utilisation des espace-temps de Malament-Hogarth. Pour illustrer ces tentatives, j'ai choisi en particulier de présenter les machines à trous noirs car elles représentent la proposition relativiste la mieux acceptée

par la communauté scientifique. Comme l'indique leur nom, ces HM utilisent les propriétés physiques de certains trous noirs en plus de celles provenant des espace-temps de Malament-Hogarth [5].

1.2.2 Les machines à trous noirs

Les ont été introduites par Etesi et Németi [Etesi and Németi, 2002]. Ces dernières sont fondées sur l'utilisation de trous noirs particuliers - appelés *trous nois de Kerr-Newman* - qui sont des trous noirs rotatifs possédant des charges éléctriques. Sans rentrer dans les détails, la particularité des trous noirs de Kerr-Newman réside dans le fait que leur partie externe constitue un *espace-temps de Kerr-Newman* qui est en fait un espace-temps de Malament-Hogarth.

Une machine à trous noirs $G = (\gamma_P, \gamma_O)$ fonctionne à l'intérieur d'un espace-temps de Kerr-Newman $\mathcal{K}$ situé à l'extérieur d'un trou noir de Kerr-Newman dont la rotation est très lente. γ_P est un ordinateur qui voyage autour du trou noir en question, tandis que γ_O est un observateur en chute libre qui se dirige vers le trou noir. Au début de l'expérience, γ_P et γ_O commencent leur course à partir du même point $q \in \mathcal{K}$. Cependant, d'après les propriétés des espace-temps de Malament-Hogarth, lorsque γ_O aura consommé une quantité de temps finie pour atteindre le point p, γ_P en aura consommé quant à lui une quantité infinie.

Les conditions de l'expérience énoncées, Etesi et Németi expliquent dans leurs travaux comment une machine à trous noirs peut vérifier l'ensemble infini des théorème de ZFC [6] afin de rendre décidable la théorie des ensembles. Plus précisément, ZFC n'est pas décidable car il n'existe pas de

5. Notons qu'il existe d'autres propositions telles que les SAD machines [Hogarth, 1994], [Hogarth, 2004]. Je n'ai toutefois pas choisi de les présenter en détails car l'étude de ces machines est essentiellement théorique ce qui n'apporte pas de nouveaux éléments à la question de leur construction physique.

6. Si la théorie des ensembles est formée uniquement à partir des axiomes qui ont été proposés par Zermelo et Fraenkel, on appelle cette théorie ZF. En revanche, si l'on rajoute à ZF un nouvel axiome nommé *l'axiome du choix*, le nouveau système est appelé ZFC. Voici un des énoncés possibles de l'axiome du choix : étant donné un ensemble A ayant pour éléments des ensembles non vides, il existe un ensemble B qui contient exactement un élément provenant de chacun des ensembles appartenant à A.

MT - ou de procédure effective - permettant de déterminer si un énoncé quelconque est un théorème de ZFC. Tout au plus, ZFC est semi-décidable au sens où les systèmes de preuves que l'on possède pour établir qu'un énoncé est un théorème de cette théorie ne fournissent de réponse que dans le cas où l'énoncé est un théorème. Plus particulièrement, si l'énoncé que l'on fournit au système est un théorème de ZFC alors une preuve de cet énoncé sera construite après un intervalle de temps fini ; mais dans le cas où l'énoncé n'est pas un théorème de ZFC, le système ne s'arrêtera jamais et l'on ne saura pas si cela veut dire (1) que l'énoncé n'est pas un théorème de ZFC ; ou (2) que le système n'a pas encore trouvé de preuve de cet énoncé.

Expliquons à présent pourquoi ZFC est décidable. Supposons que γ_P possède dans son système une MT qui construit la preuve d'un énoncé E de ZFC. De son côté, l'observateur γ_O possède lui aussi une réplique identique de cette MT qui lui permettra de recevoir un signal de γ_P et ainsi d'avoir la confirmation que l'énoncé testé est un théorème de ZFC. Lorsque γ_O commence son voyage, γ_P tente de son côté de construire une preuve de E tout en parcourant son orbite autour du trou noir. Si γ_P trouve une preuve de E alors la machine envoie un signal à γ_O qui saura donc que l'énoncé est un théorème de ZFC. Dans le cas où E n'est pas un théorème de ZFC, l'observateur ne recevra aucun message durant son trajet et lorsqu'il arrivera au point p il saura que γ_P a consommé une quantité infinie de temps. Ainsi, ZFC est décidable par machines à trous noirs.

Le résultat précédent a été formellement démontré par Etesi et Németi. Plus généralement, ces derniers ont démontré la proposition suivante :

Proposition
Soit R une relation qui est récursivement énumérable mais qui n'est pas décidable. Alors il existe une machine à trous noirs $G = (\gamma_P, \gamma_O)$ qui décide R.

Evaluons pour finir les avantages et les inconvénients de la proposition d'Etesi et de Németi. En premier lieu, la machine à trous noirs à un avantage crucial par rapport aux HM qui ont été proposées par Pitowsky d'une part et par Shagrir et Pitowsky d'autre part. Cet avantage est que l'observateur γ_O

n'entre jamais dans le trou noir proprement dit et donc qu'il peut en théorie revenir sur Terre pour communiquer le résultat fourni par γ_P. Dans le cas de l'expérience de Pitowsky, Angelene périt sous l'effet des forces gravitationnelles au moment où elle peut affirmer que la réponse au problème de Wittgenstein est négative.

Cependant, l' ne possède pas tous les avantages de celle de Shagrir et Pitowsky. Contrairement à cette dernière, l'HM d'Etesi et de Németi - et plus précisément γ_O - ne reçoit pas de signal lorsqu'elle atteint le point p ce qui peut laisser des doutes quant à la fiabilité de la réponse de γ_P. Il se pourrait par exemple que γ_P ait eu un problème lors de son voyage autour du trou noir et donc que l'absence de réponse soit due à une erreur lors de l'expérience. Quoi qu'il en soit, cet inconvénient ne remet pas en cause la possible construction physique d'une machine à trous noirs. Le problème suivant peut en revanche y parvenir.

Le principal problème allant à l'encontre de la proposition d'Etesi et de Németi est issu des travaux de Hawking [Hawking, 1975]. En 1974, Hawking propose l'hypothèse selon laquelle les trous noirs émettent des radiations thermiques appelées aujourd'hui *radiations d'Hawking*. D'après Hawking, tout trou noir qui est soumis à de telles radiations dues à des effets quantiques doit inexorablement disparaître au cours du temps. Ainsi dans le cas où l'hypothèse d'Hawking s'avère être juste, il se pourrait qu'une machine à trous noirs ne puissent jamais terminer l'exécution d'un ST puisque le trou noir en question disparaît après un intervalle de temps fini. Cet intervalle de temps dépend en particulier de la masse du trou noir. Par exemple, un trou noir possèdant la masse du soleil devrait disparaître au bout de 10^{67} années, tandis qu'un trou noir de 10^{16} kg devrait s'évaporer après 3.10^9 années.

Même si les propositions relativistes sont soumises à des problèmes théoriques et technologiques, ces problèmes ne remettent pas définitivement en cause la tentative de démontrer la TPHC à partir de ces propositions. Autrement dit, que ce soit la question d'une possible mémorisation d'une infinité de données ou celle de l'existence des espace-temps de Malament-Hogarth dans le monde réel, ces questions font encore l'objet de recherches actives. De ce point de vue, les HM relativistes sont donc plus prometteuses que leurs

homologues newtoniennes qui, comme je l'ai expliqué plus haut, ne peuvent pas être physiquement construites à cause de leurs propriétés qui sont incompatibles avec le monde réel.

Avant d'en arriver aux propositions fondées sur l'utilisation de l'aléatoire, il me reste à présenter une dernière catégorie de propositions ayant pour origine le domaine quantique.

2 Propositions quantiques

Les propositions quantiques peuvent être d'après moi classées en deux catégories. Il existe tout d'abord les propositions qui *calculent plus efficacement* que la MT mais qui n'hyper-calculent pas. L'exemple paradigmatique est le MQS théorisé par Deutsch et qui est capable de dépasser les ordinateurs actuels en termes de vitesse de calcul [Deutsch, 1985]. Viennent ensuite les propositions telles que le modèle adiabatique de Kieu qui sont conçues dans le but de démontrer la TPHC [Kieu, 2001]. Ces dernières propositions sont toutefois soumises à de sérieuses critiques à la fois théoriques et pratiques provenant de la communauté scientifique.

Paradoxalement, je pense que la première catégorie - qui inclue le modèle de Deutsch - est dans un certain sens plus prometteuse que la seconde catégorie - qui inclue le modèle de Kieu - afin de démontrer la TPHC. Cela peut paraître paradoxal car la première catégorie ne permet pour l'instant pas - même en théorie - de calculer des fonctions non Turing-calculables. Les propositions de la seconde catégorie ne sont toutefois pas logées à meilleure enseigne puisque leur possible construction est plus qu'incertaine du fait de la myriade de problèmes physiques qu'elles soulèvent. Cette section a donc un double but : celui de présenter les propositions quantiques - provevant à la fois de la première et de la seconde catégorie - et d'expliquer ma position selon laquelle le MQS pourrait laisser présager des avancées plus sérieuses que le modèle de Kieu, pourtant proposé dans le but de démontrer la TPHC.

2.1 Le modèle quantique standard

Au cours des années 1980, les possibilités calculatoires des ordinateurs classiques fondés sur la physique classique furent comparées à celles que pourraient posséder les ordinateurs s'ils étaient fondés sur la physique quantique. Les efforts initiaux menés par Deutsch permirent en particulier de concevoir un modèle de ces *ordinateurs quantiques* [Deutsch, 1985].

De nos jours, les caratéristiques de ce modèle - que l'on nomme () - sont exploitées dans le but de réduire la complexité en temps de certains problèmes. L'algorithme le plus célèbre - celui de Shor - permet par exemple de résoudre dans un temps raisonnable le problème de la décomposition d'un nombre en un produit de facteurs premiers [Shor, 1994]. Malgré ces avantages, Deutsch a néanmoins démontré que le modèle standard calcule exactement les mêmes fonctions que la MT.

Pourquoi présenter le modèle standard de l'informatique puisqu'il ne permet pas d'hyper-calculer ? Parce que selon moi, le modèle quantique fait plus que redéfinir les limites de la faisabilité pratique instaurées par la théorie de la complexité : il atteste du fait que l'on a franchi une première étape vers l'hyper-calcul.

Pour comprendre la relation qui existe entre la construction physique du modèle standard et l'hyper-calcul, je commence par exposer l'origine et les bases du calcul quantique. J'explique ensuite quelles sont les principales propriétés quantiques du modèle standard proposé par Deutsch. Enfin, je tente de détailler ma position selon laquelle les récentes avancées du calcul quantique peuvent être considérées comme une première étape en vue de démontrer la TPHC.

2.1.1 L'origine du calcul quantique

L'évolution de la puissance des ordinateurs repose sur les améliorations des semi-conducteurs. La miniaturisation de ces derniers, en particulier, a permis d'intégrer davantage de composants sur les puces électroniques, les rendant plus puissantes et plus rapides. En 1965, Moore a remarqué que depuis l'invention du circuit intégré en 1958, le nombre de composants pouvant

être intégré sur une puce avait augmenté de manière exponentielle [Moore, 1965]. Plus précisément, on observe un doublement annuel du nombre de *transistors*[7] par unité de surface. Moore prédit alors en 1975 que le nombre de composants doublera tous les 18 mois, ce que l'on dénomme abusivement *la loi de Moore*.

De nos jours, la taille des dispositifs électroniques est de l'ordre de 100 nanomètres - 10^{-9} m - et diminue de 12% par an. En théorie, cette tendance continuera pendant les quarante prochaines années avant qu'une taille ultime soit atteinte, celle correspondant à la taille d'un atome. Cependant, bien avant cette ultime limite, la taille des composants électroniques atteindra une autre limite, celle où les électrons, gouvernant les processus des semiconducteurs, commenceront à montrer que leurs comportements est gouverné par la physique quantique plutôt que par les lois de la physique classique. D'après l'*international semiconductor association roadmap*, les lois de la physique quantique gouvernent en effet les composants ayant atteint une taille de l'ordre de 30 nanomètres. La loi de Moore indique quant à elle que cette taille sera atteinte vers 2015. Par conséquent, on peut conclure que la physique quantique sera la théorie physique adéquate pour décrire le comportement des systèmes informatiques.

Cette prédiction a amené la sphère scientifique à soulever la question suivante : que se passe-t-il lorsque la physique classique échoue à décrire un ordinateur et que cette dernière est remplacée par la physique quantique ? Les discussions concernant la possibilité de concevoir un ordinateur gourverné par les lois quantiques ont débuté en 1982, lorsque Benioff a montré comment certains systèmes quantiques dépendants du temps pouvaient simuler de manière efficace les ordinateurs classiques opérant à l'aide de la logique booléenne [Benioff, 1982].

Parallèlement, Feynman s'est posé - au cours de la même année - la question opposée : est-ce qu'un ordinateur classique peut simuler efficacement un système quantique ? Ce dernier a en particulier remarqué que le nombre de variables nécessaires afin de décrire le système quantique augmentait de

7. Le transistor est le composant électronique actif fondamental en électronique utilisé principalement comme interrupteur commandé.

manière exponentielle en fonction du temps [Feynman, 1982]. Il a de plus noté que le temps requis pour exécuter le calcul augmentait de manière exponentielle en fonction du nombre de particules du système quantique. Autrement dit, si le nombre de particules est N, le temps requis pour simuler le système sera de 2^N, ce qui dépasse le nombre d'atomes de l'univers pour quelques centaines de particules. Feynman a enfin conclu qu'un ordinateur classique ne pouvait simuler un système quantique que si ce dernier ne contenait que quelques particules. Cette dernière hypothèse a ainsi ouvert la question de savoir si un ordinateur quantique pouvait dépasser les limites calculatoires des ordinateurs classiques.

Plus précisément, comparer la puissance calculatoire de deux machines peut s'effectuer sur deux niveaux : on peut les comparer en pratique en fonction de leur efficacité à exécuter des calculs ou bien les comparer en principe en fonction du nombre de fonctions mathématiques qu'ils peuvent calculer. Que se soit en pratique ou en principe, l'idée générale est de proposer un modèle théorique pour chacune des machines et de les comparer sur la base de leur efficacité et de leur capacité à calculer des fonctions mathématiques. Le modèle théorique des ordinateurs actuels est ainsi la MT, tandis que que le modèle des ordinateurs quantiques est le MQS.

2.1.2 Le modèle quantique standard

D'après le MQS, l'unité fondamentale d'un ordinateur quantique est le bit quantique appelé plus communément . Tandis qu'un bit classique peut prendre l'une ou l'autre des valeurs possibles servant à coder l'information - 0 ou 1 - un qubit a la possibilité d'être dans un état superposé de ces deux valeurs. Toutefois, cette superposition est détruite lorsqu'une mesure est effectuée, ce qui a pour conséquence de révéler une des deux valeurs classiques prises par le qubit.

On peut distinguer trois étapes principales au sein du calcul du modèle standard : la préparation de la donnée en entrée, le calcul et la mesure de la donnée en sortie. Premièrement, la préparation de la donnée en entrée et la mesure de celle en sortie doivent être exécutées de telle façon à ne pas

pertuber les autres qubits qui ne seront pas directement mesurés. La seconde étape, correspondant au calcul proprement dit, est l'étape la plus difficile à réaliser physiquement. En théorie, cette dernière est causée par l'évolution d'opérations réversibles - notion expliquée un peu plus loin - sur les qubits à condition que ces derniers soient correctement isolés de toute perturbation afin d'éviter le plus possible les effets dus à l'environnement.

Formellement, un qubit peut être dans l'état $|0\rangle$ ou dans l'état $|1\rangle$ mais il peut aussi exister dans un état superposé. Cet état est une combinaison linéaire des états $|0\rangle$ et $|1\rangle$ que l'on dénote généralement par $|\psi\rangle$. Plus précisément, une superposition d'états de $|\psi\rangle$ est notée

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

où α et β sont des *nombres complexes*[8].

Même si un qubit peut être dans une superposition d'états, il ne peut correspondre qu'aux états $|0\rangle$ ou $|1\rangle$ lorsqu'il est mesuré en pratique ; et c'est en particulier les lois de la physique quantique qui permettent de calculer la probabilité de mesurer un qubit dans l'un de ces deux différents états. Formellement,

$|\alpha|^2$ est la probabilité de trouver $|\psi\rangle$ dans l'état $|0\rangle$

$|\beta|^2$ est la probabilité de trouver $|\psi\rangle$ dans l'état $|1\rangle$

$|\alpha|$ est le module de α. Précisément, $|\alpha|^2 = (\alpha)(\alpha)^*$ où α^* est le *conjugué complexe* de α. Il est important de comprendre de quelle manière le conjugué complexe est calculé afin d'entrevoir les capacités calculatoires du modèle standard vis-à-vis de son homolgue classique.

Pour former le conjugué complexe de $z = x + iy$, le calcul quantique utilise des portes logiques pour transformer des données. Par exemple, le comportement de la porte quantique NOT qui transforme l'état $|0\rangle$ en $|1\rangle$ et inversement peut être modélisé en utilisant la représentation matricielle des qubits et de la porte NOT :

8. Les nombres complexes sont les nombres de la forme $z = x + iy$ où $i = \sqrt{-1}$.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Ainsi

$$NOT|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0*1+1*0 \\ 1*1+0*0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$NOT|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0*0+1*1 \\ 1*0+0*1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

De façon similaire, il est possible de modéliser le comportement d'autres portes logiques classiques telles que x AND $y = xy$ ou x XOR $y = x \oplus y$, où $\oplus$ est l'addition modulo 2 [McMahon, 2008].

Ce dernier résultat permet d'entrevoir pourquoi un ordinateur quantique peut simuler un ordinateur actuel. En effet, toutes les portes logiques classiques peuvent être produites à partir des portes NOT, AND et OR ; et puisque ces trois portes pouvent être simulées en retour par des portes quantiques, il suit qu'un ordinateur quantique pourrait en théorie simuler les calculs classiques. En revanche, la réciproque qui consiste à simuler un ordinateur quantique à partir de son homologue classique n'est pas aussi simple car certaines propriétés telles que la réversibilité et le parallélisme quantiques pourraient être à l'origine de la supériorité des ordinateurs quantiques.

2.1.3 Réversibilité et Parallélisme quantiques

Contrairement aux ordinateurs classiques, les ordinateurs quantiques possèdent deux propriétés qui pourraient [9] être à l'origine de leur supériorité en termes de vitesse de calcul :

9. J'utilise ici le conditionnel car la communauté scientifique n'a pas encore identifié de manière précise quelles sont les propriétés à l'origine de cette supériorité. Certains pensent en effet que la vitesse de calcul des ordinateurs quantiques proviendrait du parallélisme propre aux états quantiques, tandis que d'autres défendent que seule l'intrication quantique - propriété selon laquelle l'état quantique de deux objets doit être décrit globalement sans pouvoir séparer un objet de l'autre - est à l'origine de leur vitesse. D'autres encore soutiennent que ce sont ces deux propriétés qui garantissent leur incroyable vitesse.

1. Premièrement, le calcul quantique est *réversible* au sens où il est possible de retrouver la donnée en entrée d'un calcul à partir de la donnée en sortie. Cette propriété est particulièrement importante car elle est intimement liée aux performances des ordinateurs. En effet, plus les calculs sont réversibles et moins il y a de dissipation d'énergie [Landauer, 1961]. Cela signifie concrètement que plus un calcul est réversible, plus la chaleur des composants électroniques diminue au cours d'un calcul. Cette propriété est fondamentale car c'est essentiellement la dissipation d'énergie qui limite la vitesse des micro-processeurs. Par exemple, un processeur du XXI^{e} siècle qui dissiperait autant d'énergie que son homologue de 1950 exploserait en une fraction de seconde.

2. Deuxièmement, *le parallélisme quantique* est une des propriétés requises pour résoudre certains problèmes qui ne peuvent pas être résolus efficacement par les ordinateurs classiques. Plus précisément, étant données deux valeurs d'une fonction $f(0)$ et $f(1)$ contenues dans une superposition de l'état quantique $|\psi\rangle$, il est possible à partir du parallèlisme quantique de calculer $|\psi\rangle$ - c'est-à-dire $f(0)$ et $f(1)$ - en moins de temps que si un ordinateur classique calculait $f(0)$ et $f(1)$ séparément. Cette propriété peut être utilisée en particulier à l'aide de *l'algorithme de Deutsch* qui permet de façon imagée de *regarder les deux côtés d'une pièce de monnaie en même temps* [Delahaye, 2006].

Détaillons ces deux propriétés.

Réversibilité du calcul quantique. La réversibilité du calcul quantique comme propriété pouvant être à l'origine de la puissance calculatoire des ordinateurs quantiques soulève de nombreuses questions. Tout d'abord, qu'est-ce que la réversibilité ? Il existe en fait trois types de réversibilité :

1. La réversibilité logique caractérisée par la possibilité de retrouver les données en entrée d'un calcul à partir des données en sortie.

2. La réversibilité physique caractérisée par la possibilité de retrouver l'état initial d'un processus physique à partir de son état final.

3. La réversibilité thermodynamique caractérisée par la possibilité d'éxécuter un calcul sans dépense d'énergie et donc sans production de chaleur.

Les liens entre ces trois types de réversibilité sont très complexes et dépassent le cadre de cette thèse[10]. N'ayant pas la prétention d'expliquer de tels liens, je me limite dans ce qui suit à un seul type de réversibilité : la réversibilité logique.

Qu'est-ce qu'un calcul réversible ? Très simplement, un calcul est réversible si l'on peut retrouver la donnée en entrée du calcul à partir de sa donnée en sortie. Inversement, un calcul n'est pas réversible lorsqu'il est impossible de retrouver la donnée en entrée du calcul à partir de la donnée en sortie. Voici un exemple simple de calcul non réversible : $2 + 2 = 4$. Ce calcul n'est pas réversible car il est impossible de déterminer si la donnée en entrée est $2 + 2$ uniquement à partir de la donnée en sortie 4. La raison en est qu'il existe d'autres données en entrée telles que $3 + 1$ ou 2×2 conduisant au résultat 4.

Tout comme le précédent calcul, les calculs effectués par les ordinateurs actuels ne sont pas réversibles. Plus particulièrement, un ordinateur classique - par opposition à quantique - ne peut utiliser que des portes logiques irréversibles telle que la porte AND. La porte AND est irréversible car une même donnée en sortie peut correspondre à plusieurs données en entrée. Il est ainsi impossible de savoir si le résultat x AND $y = 0$ provient de 1 AND 0, de 0 AND 1 ou de 0 AND 0[11].

Du côté quantique, la réversibilité du calcul est en revanche logiquement possible et elle repose principalement sur une porte logique spécifique au calcul quantique : *la porte d'Hadamard* H. Cette dernière agit sur les qubits $|0\rangle$ et $|1\rangle$ de la manière suivante :

10. Les liens entre la réversibilité du calcul et la thermodynamique sont traités dans des références classiques telles que [Maxwell, 1908], [Szilàrd, 1929], [Landauer, 1961], [Bennett, 1973], [Bennett, 1985] et [Bennett, 1988] ; ou bien plus récemment dans [Machta, 1999] et [Maroney, 2009].

11. En fait, il est possible en théorie de rendre réversibles les portes logiques classiques mais la possibilité d'implémenter physiquement de telles portes est controversée [Bennett, 1973], [Fredkin and Toffoli, 1982], [Watson, 2007].

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

La principale propriété de cette porte est que deux portes d'Hadamard branchées en série - c'est-à-dire qui agissent l'une après l'autre - permet de retrouver la donnée de départ. En effet, si on applique une porte d'Hadamard à un qubit arbitraire $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, cela donne

$$\begin{aligned} H|\psi\rangle &= \alpha H|0\rangle + \beta H|1\rangle \\ &= \alpha(\frac{|0\rangle + |1\rangle}{\sqrt{2}}) + \beta(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \\ &= (\frac{\alpha+\beta}{\sqrt{2}})|0\rangle + (\frac{\alpha-\beta}{\sqrt{2}})|1\rangle \end{aligned} \tag{3.1}$$

Et si on applique une seconde fois la porte H on retrouve le résultat original [12] :

$$\begin{aligned} H[(\frac{\alpha+\beta}{\sqrt{2}})|0\rangle + (\frac{\alpha-\beta}{\sqrt{2}})|1\rangle] &= (\frac{\alpha+\beta}{\sqrt{2}})H|0\rangle + (\frac{\alpha-\beta}{\sqrt{2}})H|1\rangle \\ &= (\frac{\alpha+\beta}{\sqrt{2}})(\frac{|0\rangle + |1\rangle}{\sqrt{2}}) + (\frac{\alpha-\beta}{\sqrt{2}})(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \\ &= (\frac{\alpha+\alpha+\beta-\beta}{\sqrt{2}})|0\rangle + (\frac{\alpha-\alpha+\beta+\beta}{\sqrt{2}})|1\rangle \\ &= \alpha|0\rangle + \beta|1\rangle = |\psi\rangle \end{aligned} \tag{3.2}$$

Afin de rendre le calcul quantique réversible, l'idée est donc d'appliquer la porte d'Hadamard à des qubits de façon à retrouver la donnée en entrée du calcul. Passons à présent au parallélisme quantique.

Parallélisme quantique. La seconde propriété fondamentale du calcul quantique réside dans la possibilité de calculer les différentes valeurs d'une

12. Dans le calcul suivant, on utilise la linéarité de H, c'est-à-dire la propriété selon laquelle $H(\alpha|0\rangle + \beta|1\rangle) = \alpha H|0\rangle + \beta H|1\rangle$.

fonction simultanément. Contrairement aux ordinateurs classiques, il est en effet possible de calculer en parallèle plusieurs résultats ce qui pourrait être à l'origine de l'augmentation de la rapidité des calculs.

Toutefois, même si un état quantique peut prendre la forme d'une superposition d'états, la théorie quantique énonce que le résultat d'une mesure d'un qubit se caractérise *in fine* par l'obtention d'une valeur classique, à savoir 0 ou 1. Il semble par conséquent que la superposition des états quantiques ne soit pas une propriété suffisante pour obtenir l'ensemble des valeurs d'une fonction à l'aide d'une unique mesure.

Une solution à ce problème a été introduite par Deutsch à partir d'un algorithme quantique que l'on nomme aujourd'hui *algorithme de Deutsch* [Deutsch, 1985]. Pour comprendre - du moins intuitivement - à quoi sert l'algorithme de Deutsch, considérons le problème suivant. Si f est une fonction inconnue pouvant prendre l'une des valeurs 0 et 1, peut-on déterminer si $f(0) = f(1)$ ou si $f(0) \neq f(1)$? Avec un ordinateur classique, le problème peut être résolu en calculant successivement $f(0)$ et $f(1)$ puis en comparant les deux valeurs. A l'aide d'un ordinateur quantique, il est en revanche possible de répondre à la question en une seule opération. De façon imagée, on peut penser à la vérification d'une pièce de monnaie : a-t-elle deux faces différentes ou deux faces identiques ? L'algorithme de Deutsch permet en particulier de faire la comparaison sans regarder successivement les deux faces de la pièce. Bien sûr, cet exemple est trop élémentaire pour être d'un quelconque intérêt pratique, mais c'est l'exemple le plus simple permettant de comprendre l'algorithme de Deutsch.

Rentrons un peu plus dans les détails de l'algorithme [13]. Pour résoudre le problème qui vient d'être énoncé, l'algorithme de Deutsch consiste (1) à créer une superposition d'états quantique ; et (2) à extraire un des états au sein de la superposition.

Dans un premier temps, l'idée de Deutsch consiste à exécuter la porte d'Hadamard non plus en série mais en parallèle pour créer un état superposé de valeurs. Plus précisément, le calcul en parallèle de deux portes quantiques

13. L'algorithme de Deutsch ne sera pas expliqué formellement. Pour une présentation complète de l'algorithme, consulter [Chuang and Nielsen, 2000, p. 34].

est représenté par *le produit tensoriel* - noté $\otimes$. Le produit tensoriel permet de modéliser des états quantiques comportant de multiples qubits non isolés les uns des autres. Par exemple, si $|\psi\rangle = |\varphi\rangle \otimes |\chi\rangle$ alors $|\psi\rangle$ est dans un état superposé de $|\varphi\rangle$ et $|\chi\rangle$. Par souci de clarté, il est néanmoins préférable d'alléger la notation du produit tensoriel. Dans ce qui suit, le produit tensoriel de deux états $|\psi\rangle$ et $|\varphi\rangle$ pourra être représenté par $|\psi\rangle \otimes |\varphi\rangle$, $|\psi\rangle|\varphi\rangle$ ou même par $|\psi\varphi\rangle$ s'il n'y a pas d'ambiguité.

Illustrons à présent la superposition d'états quantique[14] à l'aide d'un exemple. Dans cet exemple, deux portes d'Hadamard sont branchées en parallèle, processus se traduisant par $H|\psi\rangle \otimes H|\varphi\rangle$. De plus, $|\psi\rangle = |1\rangle$ et $|\varphi\rangle = |1\rangle$. En sachant que $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, on a

$$\begin{aligned} H|1\rangle \otimes H|1\rangle &= (\frac{|0\rangle - |1\rangle}{\sqrt{2}})(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \\ &= \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) \end{aligned} \tag{3.3}$$

Grâce à cette procédure, on peut ainsi obtenir un état qui contient plusieurs états superposés composés des qubits $|0\rangle$ et $|1\rangle$ et donc des valeurs $f(0)$ et $f(1)$. Toutefois, un problème se pose : à partir de l'état superposé $\frac{f(0)}{\sqrt{2}} + \frac{f(1)}{\sqrt{2}}$, il est très difficile d'extraire $f(0)$ et $f(1)$. En effet, en physique quantique chaque mesure perturbe l'état superposé et gêne d'autres mesures. Il semble ainsi que l'obtention de l'état superposé $f(0)/\sqrt{2} + f(1)/\sqrt{2}$ ait été inutile puisque qu'il est impossible d'extraire à la fois $f(0)$ et $f(1)$.

Deutsch montre néanmoins dans un second temps que certaines opérations classiquement impossibles sont réalisables à partir de l'état superposé de deux variables. Plus précisément, soient a et b des variables possédant chacune deux valeurs - 0 et 1. Soit l'opération logique appelée *OR exclusif* ou *XOR* de $f(a)$ et $f(b)$ qui vaut 0 si $f(a) = f(b) = 0$ ou si $f(a) = f(b) = 1$, et qui vaut 1 autrement. Deutsch propose en particulier une manipulation de l'état

14. Il est bien évident que le parallélisme quantique est plus complexe que ce qui est exposé ici. Cependant, le but est comprendre - même schématiquement - qu'il est en principe possible d'effectuer une telle superposition. Pour de plus amples détails, le lecteur peut consulter [Le Bellac, 2008].

superposé $f(a)/\sqrt{2} + f(b)/\sqrt{2}$ qui donne le résultat suivant. Une fois sur deux, le résultat est nul : la manipulation ne donne rien, toutes les données sont perdues, mais l'expérimentateur est averti ; et une fois sur deux on obtient la bonne valeur du XOR exclusif entre $f(a)$ et $f(b)$. Cela semble très faible car jamais dans cette manipulation quantique on obtient $f(a)$ et $f(b)$ à la fois ; pire encore, on accède à la bonne valeur du XOR qu'une fois sur deux. Pourtant, cette possibilité supplémentaire en apparence anodine, permettrait à un ordinateur quantique de surpasser n'importe quel ordinateur classique actuellement construit en terme de vitesse de calcul.

Plus récemment, certains algorithmes quantiques ont été conçus pour résoudre des problèmes spécifiques ne pouvant pas être résolus efficacement par des ordinateurs classiques. En particulier, l'agorithme de Shor permet de résoudre efficacement le problème de la décomposition d'un nombre en un produit de facteurs premiers, problème qui ne peut pas être résolu efficacement par un ordinateur classique [15] [Shor, 1994]. Même si ces algorithmes ne remettent pas en cause la TPHC au sens où les problèmes résolubles par de tels algorithmes sont décidables, ils pourraient toutefois repousser certaines limites calculatoires et être considérés comme une première étape vers l'hyper-calcul.

2.2 Repousser les limites en pratique du calcul : une première étape vers l'hyper-calcul ?

Avant tout chose, je souhaite prévenir le lecteur que la position qui va être défendue - position selon laquelle le calcul quantique peut être considéré comme une première étape vers l'hyper-calcul - est avant-guardiste.

Pour la communauté scientifique, les limites du calcul peuvent être analysées de deux manières différentes : en pratique - en prenant en compte les ressources physiques utilisées lors des calculs - ou en principe - en faisant

15. On peut en outre citer l'algorithme de Grover qui permet de réduire le temps de calcul nécessaire à la recherche d'un ou de plusieurs éléments parmi un ensemble d'éléments non classés [Grover, 1996]. Cependant, même si l'algorithme de Grover permet de mener à bien cette recherche de manière plus rapide qu'un algorithme classique, le gain de vitesse est moins spectaculaire que celui octroyé par l'algorithme de Shor.

abstraction de ces ressources. J'ai en particulier longuement expliqué dans le chapitre 2 que lorsque les limites en pratique sont repousées cela n'avait pas d'influence sur les limites en principe. Les limites en principe constituent en effet une borne supérieure aux limites en pratique : puisqu'une fonction non Turing-calculable ne peut pas être calculée en principe, elle ne peut pas être calculée en pratique [16]. Plus précisément, quelle que soit la puissance de calcul d'un ordinateur - vitesse, espace mémoire - cette puissance ne lui permet pas de calculer des fonctions non Turing-calculables puisque la non calculabilité de ces fonctions a été démontré par Turing indépendamment des avancées technologiques liées aux ordinateurs. Par conséquent les limites en principe ne peuvent pas être modifiées en repoussant les limites en pratique.

Je souhaite cependant défendre ici la thèse selon laquelle une modification des limites en principe peut, dans certains cas particuliers, avoir pour origine une modification des limites en pratique.

La première partie de mon raisonnement est en accord avec la position défendue par la majorité des scientifiques. Elle consiste à montrer que s'ils venaient à être physiquement réalisés, les ordinateurs quantiques tiendraient une position charnière entre les ordinateurs classiques et les HM, à savoir entre le calcul effectif et l'hyper-calcul. Plus précisément, les résultats de Deutsch et de Shor ont pour conséquence de montrer que les ordinateurs quantiques sont supérieurs en pratique aux ordinateurs classiques mais qu'ils sont néanmoins inférieurs en principe aux HM. Le calcul quantique atteste ainsi qu'il est possible d'aller *au-delà* du calcul classique sans pour autant parvenir à l'hyper-calcul.

La seconde partie de mon raisonnement est toutefois plus originale que la première car elle consiste à défendre que la stricte opposition entre les

16. Il est toutefois nécessaire de préciser que cette dernière affirmation n'est vraie que si le calcul d'une fonction f requiert de pouvoir fournir $f(n)$ pour *tout* argument n appartenant au domaine de f. En effet, il peut être possible de calculer certaines des valeurs d'une fonction qui n'est en principe pas calculable. Par exemple, soit N un nombre supérieur à tout nombre pouvant intervenir dans un calcul en pratique faisable. Black définit la fonction f suivante : $f(n) = 2n$ si $n < N$ ou si n est le nombre de Gödel Gn d'un théorème de l'arithmétique, sinon $f(n) = 0$. La fonction f n'est pas calculable car le problème de déterminer pour tout n, si n est le Gn d'un théorème de l'arithmétique est indécidable ; mais tant que l'on peut calculer $2n$ la fonction f est en pratique calculable [Black, 2000, p. 246].

limites en pratique et les limites en principe doit être réévaluée à la lumière des résultats du calcul quantique. Je pense en effet que la position charnière occupée par les ordinateurs quantiques peut avoir des conséquences pour les limites en principe. En particulier, même si les ordinateurs quantiques ne sont pas capables d'hyper-calculer, les résultats de Deutsch et de Shor peuvent néanmoins être considérés comme des arguments en faveur de la possibilité d'une future démonstration de la TPHC.

Dans le but de comprendre en quel sens les résultats de Deutsch et de Shor peuvent servir à démontrer le TPHC, je procède en trois étapes. Tout d'abord, je montre que les ordinateurs quantiques ne sont pas des HM, c'est-à-dire qu'ils sont en-deçà de l'hyper-calcul. J'explique ensuite pourquoi les ordinateurs quantiques se situent en pratique au-delà des ordinateurs classiques. Enfin, j'apporte des arguments en faveur de la thèse selon laquelle la position charnière occupée par les ordinateurs quantiques est une première étape vers l'hyper-calcul.

2.2.1 En principe, le calcul quantique se situe en deçà des hyper-machines

Dans son article de 1985, Deutsch tente de répondre à deux questions à propos du modèle standard du calcul quantique :

1. Est-ce que le modèle standard est capable de calculer plus efficacement qu'un ordinateur classique ?
2. Est-ce que le modèle standard est capable de calculer des fonctions non calculables par MT ?

La réponse à la première question est positive tandis que la réponse à la seconde question est négative. Expliquons en détails cette seconde réponse. La question précise que l'on doit se poser est la suivante : le modèle standard du calcul quantique est-il Turing-équivalent ? Autrement dit, est-ce que les fonctions pouvant être en principe calculées par un ordinateur classique et par un ordinateur quantique sont identiques ? Si tel est le cas, cela signifie que la procédure de Deutsch ne fournie pas au modèle standard une puissance calculatoire allant au-delà de la MT.

Pour montrer que le modèle standard est Turing-équivalent, Deutsch démontre le résultat suivant : il est possible de simuler le fonctionnement du modèle standard à l'aide d'une (). Les MT que j'ai considérées jusqu'à présent sont *déterministes* au sens où à partir d'une configuration donnée, il existe au plus une configuration possible pour l'étape suivante. Une MTP est quant à elle identique à une MT déterministe excepté en ce qui concerne son programme. Dans le cas d'une MT déterministe, le programme de la machine peut être défini comme une fonction de la forme

$$Q \times S \rightarrow Q \times S \times \{R, L\}$$

qui siginifie que le programme définit pour chaque paire (état de la machine, symbole lu) au plus un seul triplet « (état suivant, symbole à écrire, direction de la tête de lecture) ». De son côté, le programme de la MTP n'est pas une fonction mais une relation Δ de la forme

$$Q \times S \subseteq Q \times S \times \{R, L\}$$

qui ne définit pas un seul triplet (état suivant, symbole à écrire, direction de la tête de lecture) mais un ensemble de tels triplets. Enoncée de manière intuitive, la MTP peut choisir n'importe lequel de ces triplets pour calculer la valeur en sortie d'une fonction appliquée à un argument donné. Une fonction $f(x_1, \ldots, x_n)$ est dite calculable par MTP si, pour chacun des arguments $(x_1, \ldots, x_n)$ du domaine de f, il existe au moins une suite de configurations exécutée par la machine conduisant à $f((x_1, \ldots, x_n))$.

Le résultat de Deutsch énonçant que le modèle standard peut être simulé par une MTP implique que le modèle standard est Turing-équivalent. En effet, toute fonction calculable par une MTP est calculable par une MT et inversement [Hopcroft et al., 1979, p. 164]. Ce résultat peut paraître surprenant mais sa démonstration est fondée sur une idée simple : il suffit que la MT simule toutes les exécutions de la MTP. Cette démarche est toutefois fastidieuse en pratique car toutes les exécutions ne peuvent pas être simulées simultanément par une MT - le ruban de la MT peut difficilement être uti-

lisé pour exécuter plusieurs exécutions simultanées. Une solution possible consiste à simuler les exécutions une par une en prenant garde de ne pas en exécuter certaines - celles pouvant être infinies. En effet, si par malchance on commence par simuler une exécution qui est infinie, on sera dans l'incapacité de passer à l'exécution suivante. La stratégie de la démonstration est donc de simuler toutes les exécutions dans l'ordre croissant en prenant en compte leurs longueurs. On commence ainsi par simuler les exécutions de longueur 1, puis celles de longueur 2 et ainsi de suite [Manna, 1974].

Le résultat de l'équivalence entre la MT et la MTP en termes de fonctions calculables atteste de la Turing-équivalence du modèle standard du calcul quantique proposé par Deutsch. Le calcul quantique est par conséquent en deçà de l'hyper-calcul - ou des HM - car il ne peut pas calculer des fonctions non calculables par MT. Pour saisir correctement la position charnière du calcul quantique au sein de l'opposition entre le calcul effectif et l'hyper-calcul, j'explique à présent pourquoi le calcul quantique permet en pratique d'aller au-delà des ordinateurs classiques.

2.2.2 En pratique, le calcul quantique se situe au-delà des ordinateurs classiques

Même si un ordinateur quantique peut en principe être simulé par un ordinateur classique, une telle simulation ne peut se faire en pratique de manière efficace. Il suit que les ordinateurs quantiques offrent un avantage en terme de vitesse de calcul contrairement à leurs analogues classiques. Plus encore, certains chercheurs soutiennent qu'il ne pourrait y avoir de progrès techniques assez significatifs pour supprimer l'écart entre la puissance calculatoire d'un ordinateur quantique et celle d'un ordinateur classique [Aaronson, 2005].

Mais que veut-on dire lorsque l'on affirme qu'un ordinateur quantique résout efficacement un problème ? La notion de *problème résoluble efficacement* est formellement définie par la théorie de la complexité qui a pour but d'établir la quantité minimale de ressources que requiert un algorithme pour résoudre un problème. D'une part, un algorithme résoud un problème de manière efficace si cet algorithme requiert une quantité *polynomiale* de

ressources, c'est-à-dire une quantité qui est majorée par la valeur d'un polynôme dont la variable est la taille des données du problème. D'autre part, un algorithme résoud un problème de manière inefficace si cet algorithme requiert une quantité *exponentielle* de ressources, c'est-à-dire une quantité qui est supérieure à la valeur de tout polynôme dont la variable est la taille des données du problème. Enfin, un problème est considéré comme étant *facile* s'il peut être résolu efficacement par un algorithme ; autrement, il est considéré comme étant *difficile*.

La notion de calcul efficace est la clé permettant de comprendre les limites en pratique du calcul. En effet, la notion de calcul efficace est couramment identifiée avec la notion intuitive de *calcul réalisable en pratique* [Papadimitriou, 1994, p. 6]. De ce point de vue, un calcul qui peut être exécuté en un temps polynomial en ressources est considéré comme étant réalisable en pratique. En revanche, un calcul qui demande un temps exponentiel pour être exécuté devra être considéré comme étant irréalisable en pratique.

L'identification de la notion de réalisabilité pratique avec celle de calcul pouvant être exécuté en temps polynomial est néanmoins sujette à controverses. Il existe en particulier des calculs efficaces qui ne sont pas exécutés par des algorithmes polynomiaux et il existe des calculs exécutés par des algorithmes polynomiaux qui ne sont pas efficaces. Par exemple, un algorithme polynomial dont la complexité est de n^{80} - où n est la taille de la donnée en entrée - sera probablement d'une valeur limitée en pratique ; tandis qu'un algorithme exponentiel dont la complexité est de $2^{\frac{n}{100}}$ pourra s'avérer utile en pratique.

Il existe cependant deux puissants arguments en faveur de la dichotomie polynomial / exponentiel. Le premier argument est d'ordre pratique. Trois types de complexité peuvent en particulier être mesurés - *cf.* chapitre 2 :

- **Une complexité au pire des cas** : complexité maximale dans le cas le plus défavorable.
- **Une complexité moyenne** : complexité à partir d'une répartition probabiliste des tailles des données [Levin, 1986].
- **Une complexité au meilleur des cas** : complexité minimale dans le cas le plus favorable.

L'analyse de l'efficacité d'un algorithme à partir de la dichotomie polynomial / exponentiel se fonde sur la complexité au pire des cas. Toutefois, lorsque dans la pratique un algorithme exponentiel se comporte de manière efficace, cela signifie généralement que les cas testés se situent principalement dans la moyenne des données pour lesquels l'algorithme est efficace. C'est pourquoi la complexité en moyenne représente plus fidèlement la pratique scientifique, à savoir les situations testées par le scientifique. L'analyse de l'efficacité à partir de la complexité au pire des cas semble donc être mal choisie. Malheureusement, le scientifique n'a que rarement accès en pratique à la distribution de la complexité d'un algorithme en fonction des données en entrée, ce qui rend de fait impossible l'utilisation de la complexité en moyenne.

Le second argument en faveur de la dichotomie polynomial / exponentiel est plus fondamental que le premier. Entre la fin des années 1960 et le début des années 1970, les chercheurs ont observé que la MT pouvait simuler efficacement les calculs pouvant être simulés par d'autres modèles. Ils ont montré plus précisément que si une fonction pouvait être calculée en k étapes par un modèle qui n'est pas la MT, alors il était toujours possible de calculer cette fonction à l'aide de la MT en $p(k)$ étapes - où p est une fonction polynomiale. Cette observation a ensuite été formulée à l'aide d'une thèse qui est la première version de ce que l'on nomme aujourd'hui la :

Thèse (Version forte de la TCT)
Tout calcul pouvant être exécuté par un modèle peut être exécuté par une MT à un facteur polynomial près.

Si la version forte de la TCT est correcte cela signifie que la MT peut être considérée comme un *mètre-étalon* de la complexité, c'est-à-dire de ce qui est en pratique calculable. Autrement dit, quel que soit le modèle que l'on considère ce dernier est simulable par une MT à un facteur polynomial près. En d'autres termes encore, si un modèle de calcul récemment conçu exécute un calcul plus rapidement qu'une MT, la différence de puissance entre ces deux modèles ne peut être que polynomiale - par opposition à exponentielle.

Malheureusement, la version forte de la TCT qui vient d'être formulée est fausse car elle est falsifiée par certains modèles différents de la MT appelés *al-*

gorithmes aléatoires. Plus précisément, Solovay et Strassen ont montré au milieu des années 1970 qu'il était possible de résoudre efficacement un problème considéré par la théorie de la complexité comme difficile : le *problème de la primalité* qui consiste à déterminer si un nombre entier est composé ou premier. Pour ce faire, Solovay et Strassen ont conçu un *test de primalité* fondé sur des processus non déterministes. Contrairement aux tests de primalité déterministes, le test de primalité de Solovay et Strassen ne détermine pas avec certitude si un nombre entier est premier ou composé. Plutôt, le test détermine avec une certaine probabilité si un nombre entier est premier ou non. En réitérant le test une quantité de fois suffisante il est néanmoins possible d'affirmer avec une très grande probabilité qu'un nombre entier est premier ou composé. Ce résultat permet en particulier de résoudre le problème de la primalité en un temps polynomial en ressources. Contrairement aux algorithmes déterministes, l'exécution d'algorithmes non déterministes peut donc résoudre efficacement des problèmes difficiles provenant de la théorie de la complexité.

La conséquence de ce résultat est que la version forte de la TCT doit être rejetée car le test de primalité peut être exécuté par une machine de Turing probabiliste (MTP). Cette machine est donc capable - contrairement à ce qu'affirme la version forte de la TCT - d'exécuter un calcul - le test de primalité - qui ne peut pas être exécuté par une MT à un facteur polynomial près. Par conséquent, la version forte de la TCT doit être reformulée à partir de la notion de MTP :

Thèse (Reformulation de la version forte de la TCT)
Tout calcul pouvant être exécuté par un modèle peut être exécuté par une MTP à un facteur polynomial près.

Tout comme pour la version forte de la TCT, sa reformulation est aujourd'hui ébranlée par les récents résultats provenant du calcul quantique et plus précisément du modèle standard de Deutsch. D'après le modèle standard du calcul quantique, il est en effet possible - en théorie - de résoudre efficacement certains problèmes qui ne peuvent pas être résolus efficacement par une MTP. Le résultat illustrant le mieux les remarquables propriétés des ordinateurs quantiques est sûrement celui de Shor montrant que le modèle standard

est capable de résoudre efficacement le problème difficile de la décomposition d'un nombre en un produit de facteur premier [Shor, 1994]. Dans le cas où les ordinateurs quantiques sont un jour physiquement construits et que le résultat de Shor est confirmé - certains essais ont déjà été ménés [Jones and Mosca, 1998] - la version forte de la TCT devra de nouveau être modifiée en remplaçant la MTP par le modèle standard du calcul quantique. Les résultats de Deutsch et de Shor renforcent ainsi fortement la thèse selon laquelle les ordinateurs quantiques sont en pratique capables d'aller au-delà des ordinateurs classiques.

En résumé, les ordinateurs quantiques occupent une position intermédiaire entre les ordinateurs classiques et les HM. En pratique, le calcul quantique va au-delà des ordinateurs classique car un ordinateur quantique peut en théorie résoudre efficacement des problèmes ne pouvant pas être résolus efficacement par des ordinateurs classiques. En principe, le calcul quantique est en deçà des HM car un ordinateur quantique n'est pas capable de calculer des fonctions non Turing-calculables.

Même si les ordinateurs quantiques ne sont pas capables d'hyper-calculer, la position intermédiaire qu'occupent ces ordinateurs permet selon moi de considérer les récents résultats du calcul quantique comme l'accomplissement d'une première étape en vue de construire physiquement une HM. Je vais ainsi tenter de défendre dans ce qui suit la thèse selon laquelle les résultats de Deutsch et de Shor sont des arguments en faveur de la possibilité d'une future démonstration de la TPHC.

2.2.3 Le calcul quantique : une première étape vers l'hyper-calcul

Comment le calcul quantique peut-il être considéré comme une première étape vers l'hyper-calcul puisque l'augmentation de la vitesse de calcul n'affecte en rien le nombre de fonctions pouvant être en principe calculées ? L'exemple des ordinateurs quantiques est frappant car même si ces derniers sont capables de résoudre efficacement des problèmes difficiles, le fait est que ces ordinateurs demeurent Turing-équivalents malgré leur remarquable puissance de calcul. Autrement dit, la vitesse de calcul n'a pas d'influence sur le

calcul de fonctions non Turing-calculables.

En fait, ce dernier point n'est pas tout à fait correct. On peut en effet considérer que l'accélération de la vitesse permette à une machine d'hyper-calculer. Il suffit pour cela de considérer la machine de Turing accélérante (MTA) - *cf.* chapitre 1. Pour rappel, une MTA a la possibilité de s'arrêter après un nombre infini d'étapes contrairement à une MT. Cette possibilité repose sur le principe suivant : le temps d'une étape de calcul est deux fois moins important que le temps de l'étape précédente. En particulier, si un calcul prend une unité de temps pour exécuter sa première itération, le temps total du calcul peut être exprimé par la série géométrique

$$\sum_{i=0}^{n} \frac{1}{2^i}$$

où i est la présente itération et n est le nombre d'itérations du calcul. Par conséquent, lorsque i tend vers l'infini, le temps total du calcul approche 2 car

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 2$$

Si le calcul peut être exécuté au moyen d'un nombre fini d'étapes, une MTA calcule exactement les mêmes fonctions que la MT. En revanche, si le calcul demande une infinité d'étapes afin d'arriver au résultat la MTA double sa vitesse à chaque étape ce qui lui permet de surpasser la puissance d'une MT. La MTA peut ainsi être considérée comme un premier argument en faveur de la thèse selon laquelle une augmentation de la vitesse de calcul - ici une augmentation non bornée - pourrait être à l'origine du calcul de fonctions non Turing-calculables. Autrement dit, la MTA est une première piste afin de montrer que des avancées technologiques liées à la vitesse de calcul pourraient avoir des conséquences sur les limites en principe.

Mais on peut aller encore plus loin en affirmant que la MTA est une extension d'un ordinateur quantique. Plus précisément, il est possible de caractériser de deux façons distinctes les différences calculatoires entre une MTA et un ordinateur quantique. La première façon - qui est celle employée le

plus fréquemment par les scientifiques - consiste à les différencier en utilisant les notions de la théorie de la calculabilité. Ainsi, l'ordinateur quantique est Turing-équivalent tandis que la MTA hyper-calcule. La seconde façon - qui est plus avant-gardiste que la première - consiste à les différencier en utilisant les notions introduites par la théorie de la complexité.

D'après cette seconde façon, la différence calculatoire entre une MTA et un ordinateur quantique n'est pas analysée en principe - en faisant abstraction des ressources physiques - mais en pratique - en prenant en compte ces ressources. Les capacités calculatoires de la MTA ne sont donc pas définies en termes de fonctions calculables mais en termes de fonctions pouvant être calculées efficacement. Prenons l'exemple du problème de la décomposition d'un nombre entier en un produit de facteurs premiers et analysons-le à partir de la MT, du MQS - modèle de Deutsch - et de la MTA :

- Pour une MT, ce problème est considéré comme difficile car sa complexité n'est pas polynomiale.
- Pour le modèle de Deutsch, ce problème est considéré comme facile puisque l'algorithme de Shor réduit la complexité du problème à un polynôme.
- Pour une MTA, ce problème est aussi considéré comme facile mais sa complexité ne prend pas la valeur d'un polynôme mais d'une constante, à savoir 2.

Cette analyse permet de tirer deux conséquences. Elle permet d'une part de montrer que le calcul de la MTA peut être étudié à partir de la notion de complexité d'un problème. Elle permet d'autre part de montrer que la complexité de certains problèmes - par exemple celui de la décomposition d'un nombre en un produit de facteurs premiers - est plus faible pour une MTA que pour le modèle de Deutsch. Ces deux conséquences méritent d'être appronfondies.

Expliquons dans un premier temps comment le calcul de la MTA peut être étudié à partir de la notion de complexité d'un problème. Plus précisément, étudier la MTA à partir de la notion de complexité d'un problème veut dire déterminer la complexité des problèmes pouvant être résolus par une MTA. A première vue, il semblerait que la complexité des problèmes - décidables ou

non - résolubles par MTA soit constante puisque cette dernière met à chaque fois 2 secondes pour parcourir l'ensemble des solutions possibles. Toutefois, cette complexité n'est selon moi pas constante mais *linéaire*. La complexité d'un problème est linéaire si le temps nécessaire pour le résoudre croît de façon linéaire en fonction du nombre de données. Autrement dit, si l'on traite deux fois plus de données, le temps d'exécution sera multiplié par deux - par exemple, la recherche séquentielle dans un tableau non trié est de complexité linéaire.

Afin de défendre mon affirmation, je fonde mon analyse sur les travaux d'Earman et Norton [Earman and Norton, 1996]. Ces derniers ont proposé une HM - nommée *machine infinie* (MI) - dont le fonctionnement est en partie similaire à celui d'une MTA. Une MI est en effet capable d'exécuter une série infinie d'opérations en un temps fini mais sa composition interne diffère de celle d'une MTA. Plus précisément, une MI est divisée en deux parties : une partie **S** - pour *slave* - et une partie **M** - pour *master*. La partie **S** a pour utilité de signaler à la partie **M** que le résultat du calcul a été trouvé. Pour cela, la partie **S** peut procéder de deux manières différentes :

1. **S** envoie un signal à **M**. Dans ce cas, cela signifie que le résultat du calcul pouvait être fourni au bout d'un nombre fini d'étapes.

2. **S** ne produit pas de signal. Dans ce cas, cela signifie que le calcul demandait un nombre infini d'étapes pour être mené à bien. Le point important est que la partie **S** sait au bout d'un intervalle de temps fini que **S** ne produira pas de signal.

Ainsi, l'avantage crucial de la MI par rapport à une MT repose sur le fait que le non-arrêt de la partie **S** n'est pas ambigu. En effet, une MT qui ne s'arrête pas peut être interprétée de deux façons : soit cette dernière n'a pas encore trouvée de solutions, soit elle ne s'arrêtera jamais. Pour une MI en revanche, le fait que **S** ne s'arrête pas n'est pas un problème car **M** pourra le savoir en un temps fini.

Pour déterminer la complexité des problèmes résolubles par MI, il est nécessaire de considérer à la fois les problèmes décidables par MT et les problèmes indécidables par MT. Pour le cas décidable - c'est-à-dire pour les

problèmes dont la solution peut être calculée après un nombre fini d'étapes - la complexité est constante. Ce résultat est aisé à démontrer puisque si c est le temps que met la MI pour exécuter une infinité d'étapes, toute solution d'un problème décidable peut être fournie après c. En revanche, la complexité relative au cas indécidable est plus difficile à déterminer.

La complexité relative au cas décidable est difficile à déterminer car contrairement au cas décidable, le temps de calcul d'une MI peut fluctuer en fonction du problème indécidable qu'elle tente de résoudre. Pour le comprendre, commençons par quelques définitions :

Définition (Forme normale prénexe)
Une formule de la logique du premier ordre est en forme normale prénexe lorsque tous ses quantificateurs sont placés en tête de la formule.

Définition (Formules purement existentielle et purement universelle)

- *Une formule purement existentielle est une formule en forme normale prénexe dont les seuls quantificateurs sont existentiels.*
- *Une formule purement universelle est une formule en forme normale prénexe dont les seuls quantificateurs sont universels.*

Définition (Ensemble décidable)
Un ensemble $A \subseteq \mathbb{N}^p$ est décidable si sa fonction caractéristique est Turing-calculable. Rappelons que la fonction caractéristique χ_A de l'ensemble A est définie par :

$\chi_A(n_1, \ldots, n_p) = 1$ *si* $(n_1, \ldots, n_p) \in A$
$\chi_A(n_1, \ldots, n_p) = 0$ *sinon.*

Définition (Prédicat décidable)
Si $P(x_1, \ldots, x_p)$ est une propriété portant sur les entiers $x_1, \ldots, x_p$ - on parle aussi de prédicat d'arité p - on dit que P est décidable si l'ensemble

$$\{(x_1, \ldots, x_p)/(x_1, \ldots, x_p) \text{ vérifie } P\}$$

est décidable.

En premier lieu, une MI décide les problèmes indécidables pouvant être représentés par des formules purement existentielles et purement universelles

où le prédicat quantifié est décidable. La partie **S** vérifie simplement toutes les valeurs que peuvent prendre les variables de la formule jusqu'à trouver un contre-exemple ou jusqu'à avoir fini de tester toutes les possibilités. Par exemple, le problème indécidable qui consiste à déterminer si une équation diophantienne possède une solution ou non est décidable par MI. En effet, déterminer si une équation diophantienne possède une solution revient à décider si une formule purement universelle. En particulier, l'équation diophantienne représentant le théorème de Fermat peut être décidé par une MI puisque ce théorème peut être retranscrit dans une formule du langage des prédicats du premier ordre purement universelle : $\forall x \forall y \forall z \forall n \ \neg F(x, y, z, n)$ avec F un prédicat décidable. Ce résultat prouve que la complexité des problèmes indécidables pouvant être représentés par des formules purement existentielles et purement universelles est égale à une constante, à savoir au temps que met la MI pour effectuer une infinité d'étapes.

Cependant, tous les problèmes indécidables par MT qui sont décidables par MI ne possèdent pas une complexité égale à une constante. En particulier, un problème possède une complexité constante s'il est décidable par une MI en ne nécessitant *qu'une seule* exécution d'une infinité d'étapes en un temps fini, à savoir d'un *supertask* (ST) - *cf.* chapitre 1. La MI telle que je viens de la présenter est en effet définie comme étant une machine pouvant exécuter un ST, par opposition à une machine pouvant en exécuter plusieurs. Il est néanmoins possible de modifier cette définition pour lui permettre d'exécuter plusieurs ST successifs. L'idée est très simple à comprendre. La MI exécute le premier ST en un temps fini, disons 2 secondes. Lorsque ces 2 secondes sont écoulées, la MI peut à nouveau éxécuter un ST en 2 secondes. Elle peut ainsi exécuter n ST en un nombre de secondes égal à $2n$.

En fait, certains problèmes qui ne peuvent pas être décrits par des formules purement existentielles ou purement universelles nécessitent plusieurs exécutions successives d'un ST pour être résolus. Considérons en particulier le *problème de la décision*. Le problème de la décision est le problème de savoir s'il existe une procédure effective capable de déterminer si une formule du premier ordre est démontrable ou non. Pour montrer que ce problème ne peut pas être décidé à partir d'une seule exécution de la MI, il suffit d'exhi-

ber une formule du premier ordre qui est indécidable par une MI ne pouvant exécuter qu'un ST.

Il est possible d'exhiber une telle formule si l'on augmente la complexité de ses quantificateurs en alternant des quantificateurs existentiels et des quantificateurs universels. Considérons par exemple la proposition qui traduit le fait qu'il existe un nombre n vérifiant une certaine relation R - décidable - pour tout nombre x : $\exists n \forall x \ R(n, x)$. Pour décider cette formule, une MI doit pouvoir vérifier de manière séquentielle chaque n. En d'autres termes, la procédure consistera à parcourir pour chaque n l'ensemble des valeurs de x, en calculant $R(n, x)$ jusqu'à ce qu'un contre-exemple soit trouvé et que la MI passe au nombre $n + 1$. Malheureusement, cette stratégie échoue car la MI ne peut exécuter qu'une seule infinité d'étapes en un temps fini, ce qui ne lui permet pas de vérifier plusieurs valeurs de x.

Cette limite intrinsèque aux MI exécutant un seul ST peut être toutefois dépassée dans le cas où l'on dispose d'une MI pouvant exécuter plusieurs ST les uns après les autres. La complexité des problèmes requérant plusieurs itérations d'un ST n'est donc pas égale à une constante mais est linéaire. Cette complexité est linéaire car le temps de calcul - représenté par le nombre de ST exécuté - croît de façon linéaire en fonction du nombre de données. Autrement dit, si l'intervalle de temps nécessaire pour exécuter un ST est égale à n secondes, la complexité de tels problèmes vaudra kn où k est le nombre de ST devant être exécuté.

Pour résumer ce qui vient d'être dit, j'ai tenté de montrer que la complexité des problèmes pouvant être décidés par une MI - ou une MTA - était au plus linéaire. Ce point était nécessaire afin de justifier les deux affirmations que j'ai énoncées plus haut, à savoir (1) que le calcul d'une MTA pouvait être étudié à partir de la théorie de la complexité ; et (2) que la complexité des problèmes résolubles par une MTA était inférieure à celle des problèmes résolubles par un ordinateur quantique. Je peux maintenant passer à la dernière partie de mon argumentation ayant pour but de défendre que les récents résultats du calcul quantique peuvent être considérés comme une étape vers l'hyper-calcul.

J'ai expliqué précédemment que la version forte de la TCT avait subi

certains changements suite aux avancées technologiques et plus précisément suite à l'augmentation de la vitesse de calcul des ordinateurs. Ainsi, la MT qui était censée caractériser ce que l'on entend par calcul réalisable en pratique a été remplacée par la MTP. De nos jours, un nouveau changement de paradigme est sur le point de s'opérer à la lumière des récents résultats du calcul quantique. Les résultats de Deutsch et de Shor permettent en particulier d'affirmer que si un ordinateur quantique venait à être physiquement réalisé, ce dernier repousserait encore une fois les limites du calcul réalisable en pratique. La version forte de la TCT s'en trouverait une fois de plus modifiée ce qui se traduirait par le remplacement de la MTP par le modèle quantique de Deutsch.

En intégrant les informations exposées ci-dessus selon lesquelles (1) la MTA peut être analysée à partir de la notion de complexité d'un problème ; et (2) que la complexité des problèmes résolubles par MTA est plus faible que celle des problèmes résolubles par le modèle quantique, il est possible d'émettre l'hypothèse suivante : peut-être les ordinateurs quantiques ne sont-ils qu'une étape vers une version forte de la TCT qui serait assez robuste pour résister à tout autre modèle qui serait physiquement construit. Autrement dit, de la même manière que les ordinateurs quantiques sont sur le point de remplacer le modèle de référence actuel - la MTP - la MTA pourrait redéfinir dans un futur non déterminé les limites en pratique érigées par les ordinateurs quantiques. La version forte de la TCT serait alors énoncée comme ceci :

Définition (Reformulation hypothétique de la version forte de la TCT) *Tout calcul pouvant être exécuté par un modèle peut être exécuté par une MTA à un facteur linéaire près.*

Dans cette optique, les progrès liés à l'augmentation de la vitesse de calcul pourraient être à l'origine d'une démonstration de la TPHC - thèse selon laquelle il possible de construire une HM. Toutefois, un tel avenir n'est même pas à portée de vue puisque les ordinateurs quantiques n'en sont qu'à leurs balbutiements. Davantage à notre portée se trouvent néanmoins certaines recherches quantiques qui sont en relation directe avec l'hyper-calcul. L'une d'entre elle concerne la construction physique d'une HM quantique fondée sur les processus adiabatiques.

2.3 Le modèle adiabatique

L'augmentation de la vitesse de calcul n'est pas le seul avantage que pourrait apporter, du moins en théorie, les propriétés de la physique quantique. La thèse élaborée par certains auteurs est qu'il pourrait exister d'autres propriétés quantiques différentes du parallélisme et de la réversibilité pouvant être utilisées dans le but d'hyper-calculer. Le principal défenseur de cette thèse se nomme Kieu, physicien dont les travaux tentent de montrer que le être à l'origine de solutions liées à des problèmes indécidables. Pour être plus précis, Kieu défend un modèle quantique fondé sur un *processus adiabatique* et qui dispose selon lui de la puissance nécessaire pour résoudre le 10^{e} problème de Hilbert - à savoir le problème de déterminer si une équation diophantienne possède une solution ou non [Kieu, 2001], [Kieu, 2002], [Kieu, 2003]. Dans ce qui suit, je présente tout d'abord le modèle adiabatique puis j'expose les objections qui ont été énoncées à l'encontre de ce modèle.

Le calcul quantique par évolution adiabatique utilise principalement un théorème de la mécanique quantique [17] : *le* [Kato, 1950]. Le théorème adiabatique décrit le comportement des solutions de l'équation de Schrödinger lorsque l'hamiltonien $H(t)$ évolue lentement par rapport au temps. De manière informelle, ce théorème affirme que si le système physique est initialement dans un état propre $|\psi(0)\rangle$ avec pour valeur propre $E(0)$, et que l'évolution est adiabatique - c'est-à-dire suffisament lente - alors l'état final du système sera proche d'un vecteur propre $|\psi(T)\rangle$ avec pour valeur propre $E(T)$. Autrement dit, le théorème adiabatique affirme que si le temps de l'évolution est suffisamment lent, l'état initial évoluera vers l'état final désiré avec une forte probabilité. Voici la procédure du calcul quantique par évolution adiabatique :

1. Le système est préparé dans l'état fondamental $|\psi(0)\rangle$ de l'hamiltonien de départ H_0.
2. Pendant un temps T, le système évolue suivant l'équation de Schrödinger avec l'hamiltonien $H(t)$. Le théorème adiabatique fixe la durée T

17. J'ai tenté de présenter - le plus clairement possible - les concepts fondamentaux de la physique quantique dans l'annexe E. Ces concepts sont notamment essentiels pour comprendre la proposition de Kieu.

qui doit être suffisament longue pour que l'évolution soit adiabatique.

3. L'état final $|\psi(T)\rangle$ est mesuré. Cette mesure est le résultat du calcul et donne avec une forte probabilité $|\psi_0(T)\rangle$.

Historiquement, le théorème adiabatique a d'abord été utilisé pour résoudre des instances du problème de la *satisfaisabilité*[18], problème qui est considéré comme difficile par la théorie de la complexité [Farhi et al., 2000]. Cette utilisation du théorème adiabatique a ensuite poussé Kieu à concevoir un *algorithme*[19] quantique capable selon lui de résoudre non pas des problèmes difficiles mais un problème indécidable par MT.

Le but de Kieu est de proposer une procédure quantique fondée sur une évolution adiabatique du système qui est capable de résoudre le problème indécidable suivant :

Définition (10^{e} problème de Hilbert)
Est-il possible de concevoir une procédure effective permettant de déterminer si une équation polynomiale à coefficients entiers une équation diophantienne - a au moins une solution ?

Pour ce faire, Kieu expose et détaille sa procédure dans trois articles [Kieu, 2001], [Kieu, 2002] et [Kieu, 2003]. Il évalue ensuite la portée de ses travaux et répond à ses détracteurs dans [Kieu and Buchdahl, 2005] et [Kieu, 2008]. Sans rentrer dans les détails de la procédure de Kieu, je vais néanmoins tenter de la présenter le plus clairement possible.

La première étape de la procédure de Kieu consiste à coder une équation diophantienne particulière en un nombre fini d'étapes dans un système ayant une infinité de niveaux d'énergie. Kieu considère plus précisément l'équation diophantienne suivante :

$$D(x, y, z) = (x + 2)^3 + y^2 - 4z^5 - 2xy = 0$$

18. Le problème de la satisfaisabilité consiste à déterminer si une formule de la logique propositionnelle arbitraire est vraie. Ce problème est évidemment décidable puisqu'il existe plusieurs algorithmes permettant de le résoudre [Nam et al., 2002]. Toutefois, aucun des algorithmes disponibles ne permet de résoudre le problème de la satisfaisabilité en un temps polynomial.

19. Il est important de noter que le terme *algorithme* employé par Kieu n'est pas adéquat. Puisque sa procédure permet de résoudre des problèmes indécidables, on peut en effet affirmer d'après la thèse de Church-Turing que cette procédure n'est pas un algorithme.

Coder cette équation afin de déterminer si elle possède au moins une solution requiert la réalisation d'un *espace de Fock*, qui est un type particulier d'espaces vectoriels infinis nommés *espaces de Hilbert*. En général, l'hamiltonien correspondant à

$$D(x_1, x_2, \ldots, x_n)$$

possède la forme suivante où l'observable est représenté sous sa forme matricielle :

$$H_P = (D(a_1^\dagger a_1, \ldots, a_m^\dagger a_m))^2$$

Ainsi, l'hamiltonien correspondant à l'équation diophantienne particulière introduite ci-dessus est

$$H_P = ((a_x^\dagger a_x + 2)^3 + (a_y^\dagger a_y)^2 - 4(a_z^\dagger a_z)^5 + 2(a_x^\dagger a_x)(a_y^\dagger a_y))^2$$

La seconde étape consiste à simuler le processus adiabatique. Enfin, si l'état fondamental peut être obtenu avec une forte probabilité, une mesure praticable en un nombre fini d'étapes est menée sur ce système. Grâce au théorème adiabatique, cette mesure extrait une information sur l'équation codée, indiquant qu'elle possède des solutions ou qu'elle n'en possède pas.

Kieu défend que son modèle adiabatique peut être physiquement réalisé à condition bien sûr de pouvoir exécuter chacune des étapes de la procédure. La majorité des critiques soulevée à l'encontre de la proposition de Kieu concerne justement la faisabilité de certaines des étapes essentielles à l'exécution de la procédure. D'une part, un nombre conséquents de problèmes physiques ont été rapportés par Hodges [Hodges, 2005], Smith [Smith, 2006] et Hagar & Korolev [Hagar and Korolev, 2007]. Ces auteurs dénoncent en particulier trois problèmes physiques liés au protocole de Kieu :

1. La capacité du scientifique à implémenter physiquement certains opérateurs hamiltoniens ayant un nombre infini de niveaux d'énergie - l'infinité de niveaux d'énergie est nécessaire pour que le système teste l'ensemble infini des entiers naturels sur l'équation codée.
2. La capacité du scientifique à obtenir physiquement les états fondamentaux - ceux qui permettent d'extraire l'information sur l'équation codée,

indiquant qu'elle possède des solutions ou qu'elle n'en possède pas.

3. Le dernier problème ne concerne pas la réalisation du protocole mais sa pertinence d'un point de vue pratique. En effet, la stratégie de Kieu est de laisser le système évoluer au cours du temps en ajoutant trois contraintes sur cette évolution : (1) l'évolution doit très lente ; (2) l'évolution doit être menée adiabatiquement sans aucun contact avec l'extérieur ; et (3) la probabilité que le système parvienne à l'état désiré doit être égale à ε, où ε est une valeur très proche de 1. La question qui se pose naturellement après avoir présenté ces trois contraintes est la suivante : à quel moment doit-on mesurer le système afin d'obtenir le résultat du calcul ? Puisque l'évolution du système doit se faire sans contact avec l'extérieur, il est en effet impossible de déterminer où en est l'évolution car toute mesure autre que la mesure finale pertubera le système. En outre, puisque la probabilité d'atteindre l'état désiré est proportionnelle au temps que met le système à évoluer, comment déterminer l'intervalle de temps qui doit s'écouler entre le début de l'évolution et la mesure ? Autrement dit, à quel moment doit-on prendre la décision de mesurer le système puisque le fait de ne pas le mesurer augmente simultanément la probabilité d'obtenir le bon résultat ?

De manière générale, la communauté scientifique émet de forts doutes quant à la réalisation physique de la proposition de Kieu. Ces doutes proviennent à la fois de problèmes pratiques tels que ceux concernant la possiblité d'implémenter certains opérateurs hamiltoniens ayant un nombre infini de niveaux d'énergie, que de problèmes théoriques tels que ceux provenant de la difficulté de déterminer mathématiquement le temps nécessaire pour parvenir à l'hamiltonien recherché.

Conclusion des deux premières sections

La conclusion générale que l'on peut déduire des deux premières sections de ce chapitre est qu'aucune des tentatives actuellement proposées - qu'elles soient newtoniennes, relativistes ou quantiques - ne parvient à fonder des bases solides pour construire physiquement une HM. Pour résumer, les pro-

positions newtoniennes telles que l'HM de Davies sont physiquement impossibles car la physique sur laquelle elles sont fondées - la physique newtonienne - ne correspond pas à la physique qui régit le monde réel. Les propositions relativistes sont quant à elles physiquement possibles - elles sont compatibles avec le monde réel - mais l'existence des espace-temps sur lesquels se fonde leur fonctionnement - à savoir les espace-temps de Malament-Hogarth - n'a jamais été démontrée expérimentalement. Les propositions quantiques telles que le modèle adiabatique de Kieu soulèvent enfin des problèmes physiques liés à la réalisation du protocole expérimental qui leur permettrait d'hyper-calculer.

Il existe cependant une dernière catégorie de propositions qui semblent être de mon point de vue particulièrement prometteuses [Calude, 2005], [Stannett, 2003]. Ces propositions sont fondées sur la notion d'aléatoire et se différencient de leurs rivales sur le point suivant : les difficultés qu'elles soulèvent sont davantages d'ordre d'ordre conceptuel que physique.

En effet, les tentatives de construction qui ont été présentées jusqu'ici ont pour principal inconvénient d'impliquer de sérieux problèmes physiques. Elles expliquent en revanche généralement correctement d'un point de vue théorique de quelles manières elles peuvent hyper-calculer. Cette situation est néanmoins inversée dans le cas des propositions fondées sur l'aléatoire car c'est en théorie et non pas en pratique que les problèmes apparaissent. Plus précisément, ces propositions sont davantages soumisent à des difficultés qui concernent leur justification d'un point de vue mathématique que leur possible fonctionnement dans le monde physique :

> Il est surprenant que nous disposions d'une méthode qui marche dans le monde physique, mais qui est soumise à des difficultés qui concernent sa justification d'un point de vue mathématique [Calude, 2005, p.12].

Le point important étant que ces difficultés liées à leur justification peuvent être selon moi surmontées.

3 L'aléatoire comme source d'hyper-calcul

Cette dernière section est consacrée à deux propositions à la fois originales et prometteuses de construction physique d'HM : les propositions de Calude [Calude, 2005] et de Stannett [Stannett, 2003].

Ces propositions sont originales car elles utilisent les propriétés de l'aléatoire pour hyper-calculer. Plus précisément, la majorité des propositions qui ont été exposées jusqu'ici reposent sur les propriétés des *machines de Turing accélérantes* (MTA) - HM capable d'exécuter des *supertasks*, à savoir des processus infinis exécutés en un temps fini. Les deux HM qui vont être présentées dans cette section sont ainsi originales au sens où elles ne sont pas fondées sur les propriétés des MTA mais sur celles des OM, c'est-à-dire d'une *machine de Turing à oracle*. Pour rappel, une OM est une MT possédant un élément supplémentaire appelé *oracle*. L'oracle a pour particularité de fournir les valeurs d'une fonction non Turing-calculable et permet de fait à la machine possédant un tel oracle d'aller au-delà de la barrière de Turing - l'OM est redéfinie en détails un peu plus bas.

Les propositions de Stannett et de Calude sont en outre prometteuses car leur élément fondamental - l'oracle - est à la fois plausible physiquement et réalisable d'un point de vue pratique. Par exemple, la stratégie de Calude consiste à fixer sur un ordinateur un appareil capable de produire une suite de nombres aléatoires *via* un processus quantique. Un tel appareil a été conçu par l'entreprise *ID Quantique* et se nomme *Quantis*[20].

Si la création d'un générateur de suites aléatoires tel que Quantis est physiquement réalisable, l'utilisation de l'aléatoire comme source d'hyper-calcul ne fait cependant pas l'unanimité et certains chercheurs remettent en cause la capacité des HM de Calude et de Stannett à calculer une fonction non Turing-calculable. Ces doutes vis-à-vis de l'aléatoire en tant que source d'hyper-calcul proviennent principalement de trois difficultés liées (1) à la définition d'une suite de nombres aléatoires ; (2) à la production d'une telle suite ; et (3) à la vérification du caractère aléatoire de la suite. Malgré ces difficultés, je pense tout de même que les propositions de Calude et de

20. `http://www.idquantique.com/`

Stannett sont prometteuses notamment parce que les deux premières difficultés peuvent être surmontées. La troisième, en revanche, pose davantage de problèmes c'est pourquoi elle sera traitée dans le prochain chapitre.

Afin de défendre les HM fondées sur l'aléatoire, je vais tout d'abord essayer de lever la première difficulté qui vient d'être énoncée en clarifiant le concept de suite de nombres aléatoires. Je présenterai ensuite les stratégies de Calude et de Stannett ce qui m'amènera à expliquer de quelle manière on peut surmonter la seconde difficulté.

3.1 Qu'est-ce qu'une suite de nombres aléatoires ?

Définir précisément ce qu'est une suite de nombres aléatoires () nécessite de fournir une définition formelle ou mathématique de cette notion. En effet, la principale raison pour laquelle il est indispensable de formaliser la notion de SNA réside dans le fait que notre intuition est généralement incapable de différencier une suite aléatoire d'une suite non aléatoire. Ce point peut être illustré à l'aide d'un exemple simple. Supposons que l'on définisse intuitivement une SNA comme une suite de nombres n'étant gouvernée par aucune règle. En acceptant cette définition, on serait ainsi tenté de dire que la suite 01101010000010011110011 est aléatoire - la suite ne semble être gourvernée par aucune règle. Cette suite peut néanmoins être produite à partir des 23 premiers chiffres de la décomposition binaire de $\sqrt{2}-1$. Elle n'est donc pas aléatoire car il existe une règle permettant de calculer $\sqrt{2}-1$ et de produire la suite en question.

Afin de proposer une définition formelle d'une SNA et de garantir la simplicité ainsi que la généralité d'une telle entreprise, les mathématiciens ont procédé à deux idéalisations. D'une part, ces derniers se sont limités aux suites de 0 et de 1 afin de simplifier le cadre formel. De telles suites peuvent être vues comme une abstraction d'une répétition de lancer à pile ou face avec une pièce équilibrée, c'est-à-dire une pièce pour laquelle la probabilité d'obtenir pile ou face est égale à 1/2. En associant les valeurs 1 à pile et 0 à face, on peut associer une suite binaire de 0 et de 1 à tous les résultats possibles de cette expérience - répétitions du lancer. D'autre part, les mathématiciens

n'ont pas restreint leur théorie aux suites finies mais ont considéré dans un souci de généralité toutes les suites infinies de 0 et de 1.

Dans ce qui suit, je commmence par présenter les principales formalisations de la notion de SNA. Je montre ensuite que ces formalisations convergent vers une définition unanimement acceptée. En particulier, cette convergence se cristallise en une thèse nommée *thèse de Martin-Löf-Chaitin* qui entretient des relations étroites avec la thèse de Church-Turing.

3.1.1 L'aléatoire comme absence de régularité

On serait tenté à première vue de dire qu'une suite est aléatoire si elle ne présente aucune régularité non triviale. D'un point de vue formel, cela signifie qu'une suite de 0 et de 1 est aléatoire s'il n'existe aucun théorème mathématique exhibant une régularité non triviale à propos de cette suite. Cependant, si l'on se fonde sur cette définition qui semble séduisante, aucune suite ne pourrait être qualifiée de SNA. Calude rappelle en effet que van der Waerden a énoncé un théorème indiquant qu'il existe une régularité non triviale partagée par toute suite de 0 et de 1 [Calude, 2000, p. 5] :

Théorème (Van der Waerden)
Dans toute suite binaire - infinie - au moins un des deux symboles doit apparaître dans des progressions arithmétiques de toutes longueurs.

Expliquons ce théorème. Les progressions arithmétiques ont un comportement qui n'a rien d'erratique et qui, bien au contraire, est très ordonné. Elles sont la régularité même. Une progression ou suite arithmétique est une suite - finie ou infinie - de nombres entiers $a_1, a_2, \ldots, a_n, \ldots$ telle que la différence entre deux termes successifs est constante ; c'est-à-dire s'il existe un entier r - appelé *raison de la suite* - tel que $a_{n+1} - a_n = r$ pour tout n. Dans le cas d'une suite finie $a_1, a_2, \ldots, a_n$, on appelle n la *longueur* de la suite. Le théorème de Van der Waerden énonce par conséquent que dans chaque suite binaire infinie, au moins un des deux symboles - 0 ou 1 - doit apparaître dans des suites arithmétiques de toutes longueurs.

Même si ce résultat montre que la notion de régularité ne peut pas être utilisée de cette manière pour définir une SNA, notons toutefois que la preuve du

théorème de van der Waerden n'est pas constructive. Une *preuve constructive* est une preuve mathématique qui respecte les contraintes des *mathématiques intuitionnistes*. En particulier, une de ces contraintes est de ne pas faire appel au principe du tiers exclu. Par exemple, démontrer l'impossibilité de l'inexistence d'un objet ne constitue pas une démonstration constructive de son existence : il faut pour cela en exhiber un, expliquer comment le construire en construisant un algorithme. La preuve du théorème de van der Waerden n'est justement pas constructive car il n'existe pas d'algorithme - ou de MT - qui peut déterminer en un temps fini lequel des deux symboles apparaît dans des progressions arithmétiques de toutes longueurs. En effet, puisque la suite binaire est de longueur infinie une MT ne peut pas en un temps fini parcourir la totalité de la suite pour vérifier le théorème.

Le caractère non constructif du résultat de van der Waerden permet en outre de proposer une définition alternative d'une SNA : une suite est aléatoire s'il n'existe pas de régularité constructive, c'est-à-dire s'il n'existe pas d'algorithme, de MT ou de règles de formation permettant de construire la suite. En ce sens, une suite serait aléatoire dans le cas où il n'existerait pas de MT capable de produire l'ensemble des symboles qui la constitue.

Cette nouvelle définition est particulièrement élégante car elle identifie les suites aléatoires définies sur l'ensemble $\{0, 1\}$ avec les fonctions non calculables par MT définies sur $\{0, 1\}$. Malheureusement, il est possible de montrer à l'aide d'un argument basé sur la cardinalité que l'absence de régularité calculable n'est pas une condition adéquate pour définir la notion de SNA. Premièrement, puisqu'il existe un nombre indénombrable de fonctions non Turing-calculables [21] il existe aussi un nombre indénombrable de suites aléatoires. Deuxièmement, considérons l'ensemble des suites telles que $\forall n \geq 1, x_{2n} = x_{2n+1}$, où l'indice d'un symbole renvoie à la position de ce symbole au sein de la suite. De telles suites sont trop ordonnées *localement* pour être aléatoires, mais puisqu'il en existe un nombre non dénombrable, certaines d'entres elles doivent être des suites aléatoires [Volchan, 2009]. Par

21. Rappel : puisque l'ensemble des fonctions définies sur $\{0, 1\}$ est indénombrable et qu'il n'existe qu'un nombre dénombrable de MT, l'ensemble des fonctions non Turing-calculables est indénombrable.

conséquent l'ensemble des suites non Turing-calculables est trop large pour caractériser l'ensemble des suites aléatoires, ce qui rend l'implication suivante fausse : si une suite n'est pas Turing-calculable alors elle est aléatoire. En résumé, l'absence de régularité échoue à caractériser la notion de SNA.

3.1.2 L'approche stochastique de Von Mises

La première tentative afin de définir la notion de SNA à partir de propriétés stochastiques fondées sur les probabilités fut détaillée par von Mises [von Mises, 1941] [von Mises, 1957]. Voici une présentation informelle de cette approche.

Présentation. Considérons une sous-suite quelconque $x_1, \ldots, x_{n-1}$ d'une suite - les éléments de cette sous-suite ne sont pas nécessairement consécutifs - et supposons que l'on souhaite prédire la valeur de x_n. Si la suite était réellement aléatoire alors aucune des deux informations suivantes ne devrait être d'une quelconque utilité (1) les valeurs d'un des membres de la suite ; et (2) la place au sein de la suite de la valeur que l'on souhaite prédire. Dans le cas contraire il existerait une régularité exploitable au sein de la suite aléatoire et un joueur pourrait, par exemple, parier sur son résultat préféré et être assuré d'obtenir un gain positif à l'aide de ces informations.

Un *système de jeu* (*gambling system*) sélectionne des points dans une suite de résultats pour que l'on parie dessus, et un *système de jeu performant* (*successful gambling system*) est un système sélectionnant des points qui ont une probabilité plus grande d'être des points gagnants que s'ils étaient choisis au hasard. Autrement dit, utiliser un tel système doit permettre d'obtenir de meilleurs résultats que si l'on se repose uniquement sur la chance. C'est l'observation empirique des systèmes de jeu qui a poussé von Mises à définir la notion de SNA. Constatant qu'il n'existait pas - au sens propre - de système de jeu performant, von Mises proposa la définition suivante : une suite infinie de nombres composée uniquement de 0 et de 1 est aléatoire si elle ne peut pas être exploitée par un système de jeu performant.

D'après von Mises, puisqu'il n'existe pas de système qui sélectionne un résultat sur la base des résultats précédents, une suite aléatoire serait une

suite pour laquelle aucune sous-suite $x_1, \ldots, x_{k-1}$ ne fournit d'informations concernant le résultat x_k. Cette idée est précisée dans un premier temps par l'auteur à partir de la notion de *règle de sélection - place selection* - qui est

> Le fait de sélectionner une suite partielle de telle façon que nous décidons si un élément devrait ou ne devrait pas être inclus sans faire usage de la valeur de l'élément [von Mises, 1957, p. 25].

L'auteur définit dans un second temps une suite infinie aléatoire comme ceci : soit p_0 et p_1 les probabilités des symboles 0 et 1 au sein de la suite infinie ; cette suite est aléatoire si chaque sous-suite infinie sélectionnée par *une règle de sélection admissible* conserve les probabilités p_0 et p_1. Autrement dit, cela signifie que toutes les sous-suites sélectionnées à l'aide d'une règle de sélection admissible doivent satisfaire la *loi des grands nombres* - qui est définie plus bas - avec une probabilité égale pour les symboles 0 et 1.

Formellement, une suite binaire infinie $x = x_1, x_2 \ldots$ est une SNA pour von Mises si et seulement si elle satisfait les deux propriétés suivantes :

I. Soit $f_n = \sharp\{m \leq n : x_m = 1\}$ le nombre de 1 parmi les n premiers termes de la suite x. Alors

$$\lim_{n\to\infty} \frac{f_n}{n} = p$$

existe et $0 < p < 1$.

II. Si $\Phi : \{0,1\}^* \longrightarrow \{0,1\}$ est une fonction partielle admissible - à savoir une règle de sélection pour une sous-suite de x tel que x_n est choisi précisément lorsque $\Phi(x_1 x_2 \ldots x_{n-1}) = 1$ - alors la sous-suite $x_{n_1} x_{n_2}$ ainsi obtenue satisfait la propriété **I** pour le même p.

La propriété **I** est un théorème de la théorie des probabilités appelée *loi des grands nombres*. Intuitivement, cette propriété décrit la probabilité d'obtenir un certain résultat lors d'une expérience qui est exécutée un grand nombre de fois. Plus précisément, la moyenne des résultats obtenus à partir d'un grand nombre d'exécutions empiriques de l'expérience - appelées *épreuves* - devra être très proche de la valeur théorique attendue, et cette différence entre les résultats empiriques et théoriques se fera de plus en plus petite à mesure que le nombre d'épreuves augmentera. Par exemple dans

notre cas où les résultats 0 et 1 de la suite sont considérés comme les résultats d'un lancer de pièce équilibrée, la probabilité théorique que le résultat soit 0 ou 1 est égale à 1/2. D'après le théorème, la proportion de 1 après n lancers devra ainsi converger vers 1/2 lorsque n tend vers l'infini.

Toutefois, la propriété **I** est clairement insuffisante pour définir une SNA et elle doit être complétée par la propriété **II**. En effet, la propriété **II** stipule que les probabilités des symboles 0 et 1 doivent rester identiques pour toute sous-suite infinie que l'on formerait à partir d'une règle de sélection admissible. Cette propriété est nécessaire afin de rejeter les suites de nombres telles que 0101010101... qui satisfont la loi des grands nombres dans le cas de la suite prise comme un tout, mais qui ne préservent pas les résultats obtenus à partir de cette loi pour certaines sous-suites infinies sélectionnées à l'aide d'une règle de sélection admissible - prendre par exemple la sous-suite $S(x) = 2n$ où n est le symbole de la suite à la n-ième place.

Critiques. L'approche stochastique de von Mises a subi de nombreuses critiques. Ces critiques proviennent tout d'abord du caractère imprécis ou semi-mathématique de sa définition. Précisément, la notion centrale de règle de sélection admissible n'est pas claire du tout. D'une part, quel type de règle de sélection doit être condidéré comme admissible ? Une règle fondée sur une loi arithmétique telle que *sélectionner* x_n *tel que* n *est premier* ? Ou bien une règle qui dépend des éléments de la suite que l'on étudie, règle qui pourrait être *sélectionner la suite dès que la première occurence de* 001 *apparaît* ? D'autre part, si l'on considère toute règle comme étant admissible, on arrive à un problème qui fut énoncé par Kamke [Kamke, 1932]. Etant donnée une suite $\{n_k\}_{k\geq 1}$ avec $n_1 < n_2 < \ldots$, considérons une sélection à partir de x de la sous-suite $x_{n_1}, x_{n_2} \ldots$ D'après Kamke, si toutes les sous-suites sont permises alors il en existera une telle que $\forall k, x_{n_k} = 1$ et une autre telle que $\forall k, x_{n_k} = 0$. Par conséquent aucune suite x ne peut être aléatoire si toutes les règles de sélection sont permises.

La réponse de von Mises fut de dire que la preuve de Kamke n'est pas constructive car aucune règle n'est mentionnée afin de construire les sous-suites dont ce dernier prouve l'existence. C'est pourquoi Alonzo Church pro-

posa l'idée de resteindre les règles admissibles - qui sont des fonctions partielles - aux fonctions partielles calculables [Church, 1940].

Dans un premier temps, la proposition de Church a été considérée comme adéquate pour caractériser une SNA :

1. Elle permet tout d'abord de rester fidèle à l'intuition selon laquelle une SNA ne peut pas être construite à partir d'un algorithme ou d'une MT. Dans le cas contraire, on pourrait en effet construire deux règles de sélection calculables Φ_0 et Φ_1 - deux fonctions partielles calculables - de la manière suivante :

$$\begin{cases} \Phi_0(x_1x_2\ldots x_{n-1}) = 1 \text{ si } x_n = 0, \\ \Phi_1(x_1x_2\ldots x_{n-1}) = 0 \text{ si } x_n = 1. \end{cases}$$

2. Elle permet ensuite de prouver que chaque SNA est *Borel-normal* en base 2, c'est-à-dire que chaque sous-suite de longueur k au sein de la suite x apparaît en suivant une probabilité asymptotique de $1/2^k$. Ce résultat montre que les définitions de von Mises et de Church sont mathématiquement robustes car elles satisfont la Borel-normalité qui est une propriété que doit partager toute SNA. Cette propriété n'est maleureusement pas suffisante, car il existe des nombres Borel-normaux qui ne sont pas aléatoires. Par exemple, le nombre de Copeland-Erdös $0,23571113171923\ldots$ obtenu par concaténation des nombres premiers est à la fois Borel-normal et non aléatoire [Copeland and Erdös, 1946].

Dans un second temps cependant, Jean Ville a montré en 1939 que les suites aléatoires définies par von Mises n'étaient pas *assez aléatoires* car il existait des suites satisfaisant la définition formelle de von Mises mais pour lesquelles

$$\frac{f_n}{n} \geq \frac{1}{2}$$

pour tout n. Ce résultat, montrant que certaines suites laissaient apparaître une préférence pour le symbole 1, a été considéré comme un coup fatal envers l'approche stochastique de von Mises.

3.1.3 L'incompressibilité de Chaitin-Kolmogorov

Même si la définition de von Mises est inadéquate pour caractériser la notion de SNA, ses travaux ont été fondamentaux pour la recherche d'une telle définition. En effet, plusieurs définitions robustes ont été proposées peu de temps après. Parmi ces définitions figurent en particulier celles de Chaitin-Kolmogorov, de Schnorr et de Martin-Löf. Je présente dans ce qui suit ces trois définitions, bien qu'une attention toute particulière soit apportée à celle de Chaitin-Kolmogorov, cela en raison du fait que la définition de Chaitin-Kolmogorov utilise des concepts qui ont été préalablement étudiés tout au long de ce travail.

L'idée de définir une SNA comme une suite qui est *incompressible* fut proposée indépendamment par Solomonoff [Solomonoff, 1964], Kolmogorov [Kolmogorov, 1965] et Chaitin [Chaitin, 1966]. Contrairement à la tentative de von Mises qui est fondée sur des propriétés stochastiques provenant de la théorie des probabilités, la théorie de Chaitin et Kolmogorov repose sur des notions issues de l'informatique théorique. Voici les propriétés fondamentales de cette théorie.

Présentation. L'intuition qui se cache derrière la théorie de Chaitin-Kolmogorov est qu'une suite de nombres est aléatoire, c'est-à-dire *irrégulière* ou *sans comportement remarquable*, s'il n'est pas possible de la générer - la décrire - plus efficacement qu'en fournissant l'ensemble de la suite. En d'autres termes, une suite de nombres x est aléatoire si aucun programme informatique d'une taille - longueur - plus petite que x ne peut générer - décrire - x.

Expliquons davantage cette idée. Partant de l'intuition selon laquelle une suite de nombre est aléatoire si elle ne présente aucune régularité, Chaitin et Kolmogorov ont proposés la notion informelle de *description efficace* : un nombre peut être décrit efficacement si sa nouvelle description utilise moins de symboles que pour sa précédente description. Par exemple, certains nombres en base 10 tels que 1,000,000,000,000 peuvent être représentés efficacement par 10^{12} ; ils présentent donc une régularité et ne sont pas aléatoires. En revanche il sera très difficile de trouver une description efficace d'un

nombre tel que $5,172,893,164,583$ à moins de le décire symbole après symbole. De tels nombres ne présentant pas de régularité auront plus de chances d'être considérés - conformément à l'intuition - comme étant aléatoires.

Plus précisément, une suite qui présente des régularités pourra être décrite efficacement par un algorithme ou programme. Pour s'en convaincre, prenons le nombre binaire $1111\ldots1111$ composé de 10000 occurences du symbole 1. Ce dernier peut être décrit efficacement par le programme suivant : *écrire le symbole 1, 10000 fois*. En revanche, si l'on considère une suite arbitraire composée de 10000 symboles qui ne présente aucune régularité apparente, il est probable que le seul moyen permettant de générer cette suite soit d'inscrire un par un les symboles qui la constituent. En d'autres termes, une telle suite est si désordonnée qu'elle ne peut pas être décrite à partir d'une description contenant un nombre de symboles inférieur à la longueur de la suite.

Afin de formaliser ces notions, considérons $w \in \{0,1\}^*$ et prenons U comme étant une MT universelle. De plus, soient $U(p)$ la donnée en sortie de la machine U qui travaille sur l'entrée p définie sur $\{0,1\}^*$ et $|p|$ la longueur - en bits - du mot p. Definissons tout d'abord la *complexité algorithmique* d'un mot relativement à une MT :

Définition (Complexité algorithmique)
La complexité algorithmique $K_U(w)$ d'un mot w relative à la MT U est donnée par

$$K_U(w) = \begin{cases} \infty \text{ s'il n'existe aucun } p \text{ tel que } U(p) = w, \\ \min\{|p| : U(p) = w\} \text{ autrement.} \end{cases}$$

Pour le dire autrement, $K_U(w)$ est la taille du plus petit programme p qui permet à U de générer w en sortie et de s'arrêter. C'est une façon de dire que $K_U(w)$ est la taille du plus petit programme qui décrit w. Le point important est que $K_U(w)$ est universelle c'est-à-dire que cette complexité algorithmique est indépendante de la MT universelle que l'on utilise :

Théorème (Théorème d'invariance)
Si U est une MT, alors pour toute MT universelle U' on a

$$K_U(w) \leq K_{U'} + c_{U'}$$

pour tout $w \in \{0,1\}^*$*, où* $c_{U'}$ *est une constante indépendante de* w*.*

La proposition suivant montre qu'il n'existe que peu de mots ayant une faible complexité :

Proposition
$\sharp\{w \in \Sigma^* : K(w) < k\} < 2^k$.

Preuve : on peut énumérer par ordre croissant tous les programmes de longueur inférieure à k :

$$\Lambda, 0, 1, 00, 01, 10, 11, \ldots, 111\ldots11,$$

ce qui conduit à un total de $1+2+4+\ldots+2^{k-1} = 2^k-1$ programmes. Puisque chaque programme produit au plus une donnée en sortie pour chaque donnée en entrée, on obtient le résultat établi dans la proposition.

Enfin, Kolmogorov formule l'idée selon laquelle une suite infinie est aléatoire si cette suite est composée de segments initiaux ayant une forte complexité. De telles suites sont dites *incompressibles*.

Définition (Suites incompressibles)
Une suite x *de* $\Sigma^{\mathbb{N}}$ *est incompressible lorsqu'il existe une constante* c *telle que*

$$K(x(n)) \geq n - c$$

pour tout n*, où* $x(n) = x_1x_2\ldots x_n$*.*

Critiques. Malgré la force de la défition de Chaitin-Kolmogorov, Martin-Löf a montré qu'il n'existait pas de suite incompressible au sens de Kolmogorov :

Théorème (Martin-Löf)
Si $f : \mathbb{N} \to \mathbb{N}$ *est une fonction calculable telle que*

$$\sum_{n=1}^{\infty} 2^{-f(n)} = \infty,$$

alors pour toute suite binaire $x = x_1x_2\ldots$ *on a*

$$K(x(n)) < n - f(n)$$

pour une infinité de valeurs de n.

En particulier ce théorème est vrai pour $f(n) = \log_2 n$. Par conséquent la complexité algorithmique de chaque suite binaire baisse infiniment souvent en-dessous de $n - \log_2 n$, c'est-à-dire bien en-dessous de sa taille.

Après ce coup décisif porté par Martin-Löf, Chaitin a néanmoins montré que la notion de suite incompressible pouvait être sauvée si on se limitait à une certaine classe de MT : *les machines de Turing à préfixe*. La principale propriété qui a poussé Chaitin à modifier la notion de Kolmogorov à partir des machines de Turing à préfixe est la suivante. Supposons que la suite p est la description - le programme - d'un mot w par une MT. Dans le cas où l'on fournit p à une MT qui n'est pas à préfixe, sa tête de lecture devra obligatoirement commencer par scanner chaque symbole de p afin d'obtenir $|p| = n$ - sa taille ; ce n'est qu'après qu'elle pourra commencer à calculer à partir de p. Ainsi la complexité de w pourrait bien être égale à $n + \log_2 n$ au lieu de n. Cette dernière possibilité ne peut cependant pas arriver dans le cas d'une machine de Turing à préfixe car par construction le programme p sera scanné uniquement vers la droite, et la machine s'arrêtera lorsqu'elle aura atteint le dernier symbole de p.

On peut à présent terminer par donner les deux définitions de Chaitin qui remplacent celles de Kolmogorov :

Définition (Complexité algorithmique de Chaitin)
La complexité algorithmique de Chaitin $C_U(w)$ *d'un mot* w *relative à la machine de Turing à préfixe* U *est donnée par*

$$C_U(w) = min\{|p| : U(p, \Lambda) = w\}$$

Définition (Suites incompressibles)
Une suite x *de* $\Sigma^{\mathbb{N}}$ *est incompressible lorsqu'il existe une constante* c *telle que*

$$C(x(n)) \geq n - c$$

pour tout n, *où* $x(n) = x_1 x_2 \dots x_n$.

Pour terminer de caractériser la notion de SNA, présentons les définitions qui ont été apportées par Schnorr et Martin-Löf.

3.1.4 Martinguales de Schnorr et tests statistiques de Martin-Löf

Dans ce qui suit, je ne présente pas en détails les approches de Schnorr et de Martin-Löf. La raison en est que pour être parfaitement comprises ces deux dernières approches requièrent des outils mathématiques complexes qui dépassent le cadre de ce travail. J'essaie néanmoins de présenter les martingales de Schnorr et les tests statistiques de Martin-Löf avec le plus de justesse possible.

Les martingales de Schnorr. L'approche de Schnorr est assez similaire à celle de von Mises au sens où elle se fonde sur la notion de *stratégie de jeu* ou de *martingale* [Schnorr, 1971]. Pour comprendre de manière intuitive sa définition d'une SNA, prenons l'exemple des jeux de paris.

Dans tout jeu de paris et à chaque fois que l'on mise, il existe une probabilité p de gagner et une probabilité q de perdre. Supposons que je joue à un jeu de paris identique au jeu *pile ou face*. Si la position de la pièce après le lancer est celle sur laquelle j'ai misé, je gagne le double de ma somme de départ, sinon je perds ma mise. Supposons aussi que la somme de départ que je possède ou celle que mon adversaire possède puisse être finie ou infinie. Supposons enfin que la probabilité de gagner puisse changer, autrement dit supposons que p puisse être supérieure à $1/2$ ou inférieure à $1/2$.

Lorsque le jeu est en ma faveur - p est supérieure à $1/2$- la probabilité que je devienne riche est forte, et cela indépendamment du fait que ma somme de départ soit finie ou que celle de mon adversaire soit infinie. De même, je deviendrai encore plus riche avec une forte probabilité si je commence avec une somme d'argent infinie. Au contraire, si p est inférieure à $1/2$ et que ma somme de départ est finie, alors la probabilité que je perde l'intégralité de ma somme de départ est forte. Mais que se passe-t-il si p est inférieure à $1/2$ et que ma somme de départ est infnie ? Il existe dans ce cas une *stratégie de jeu*

ou *martingale* qui me permet de gagner malgré le fait que le jeu soit en ma défaveur. En effet, la martingale que l'on nomme *martingale géométrique* me permet de gagner. Cette martingale consiste à doubler la mise K à chaque fois que je perds jusqu'au moment où je gagne. Plus précisément, lorsque que gagne je perds $K + 2K + 4K + \ldots + 2^n K$ mais je gagne $2^{n+1}K$, ce qui revient à gagner K.

La notion de martingale telle qu'elle vient d'être illustrée a été reprise par Schnorr pour définir ce qu'est une SNA. Pour Schnorr, une suite est aléatoire s'il n'existe aucune martingale qui permet de gagner lorsque l'on parie sur le prochain chiffre de la suite. Formellement, une suite s est aléatoire s'il n'existe aucune martingale M telle que $\lim \sup_n G(s_n) = \infty$, où G est le gain et s_n les segments initiaux de la suite s. On retrouve ici la notion de règle d'admission introduite par von Mises et l'idée qu'une suite est aléatoire lorsque l'on ne peut pas gagner à coup sûr en sa présence.

Les tests statistiques de Martin-Löf. Si les nécessitent l'appareil mathématique le plus complexe, c'est néanmoins la définition qui est la plus utilisée au sein de la communauté scientifique [Martin-Löf, 1966]. Plus précisément, Martin-Löf considère l'univers de Cantor 2^ω qui contient toutes les suites binaires infinies ; le point important étant ici que seules les suites infinies sont considérées. Il définit ensuite les SNA de la manière suivante :

Définition

Une suite $s \in 2^\omega$ est aléatoire si et seulement si elle surmonte tous les tests statistiques effectifs.

Intuitivement, on peut considérer un test statistique comme une propriété que doit posséder une suite s pour être aléatoire. Voici des exemples de telles propriétés :

- La suite s doit être un *nombre transcendant*, à savoir un nombre réel ou complexe qui n'est racine d'aucune équation polynomiale $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^1 + a_0 = 0$, où $n \geq 1$ et $a_i \in \mathbb{N}$.
- La suite s doit être un *nombre normal*, à savoir un nombre réel tel que la fréquence d'apparition de tout n-uplet dans la suite de ses décimales dans toute base est équirépartie [Delahaye, 1997].

– La suite s doit suivre la *loi des grands nombres* qui énonce que les symboles 0 et 1 de la suite doivent être considérés comme les résultats d'un lancer de pièce équilibrée où la probabilité théorique respective des résultats 0 et 1 est égale à 1/2. Ainsi, la proportion de 1 après n lancer devra converger vers 1/2 lorsque n tend vers l'infini.

Martin-Löf prouve en outre deux résultats à propos des tests statistiques effectifs :

1. Il existe un nombre infini dénombrable de tests statistiques effectifs. Ce résultat implique que la plupart des suites binaires infinies sont aléatoires.

2. Il existe un *test universel* tel que si une suite surmonte ce test alors elle surmonte tous les autres tests statistiques effectifs. Toutefois, Martin-Löf a montré que l'on ne peut pas avoir accès à ce test et donc qu'il ne peut être utilisé afin de démontrer qu'une suite est aléatoire.

On comprend maintenant très bien le problème inhérent à toute suite de nombres que l'on supposerait être aléatoire : il est impossible de prouver qu'elle est effectivement aléatoire. Pour le prouver, il faudrait en effet appliquer un nombre infini dénombrable de tests statistiques à cette suite - ce qui est en pratique impossible - ou appliquer un test universel qui est inaccessible. Dans la pratique scientifique, on considère par conséquent une suite comme étant aléatoire lorsqu'elle a surmonté un ensemble de tests statistiques que l'on juge significatifs.

Toutefois, même si l'on considère une suite comme étant aléatoire lorsqu'elle surmonte un ensemble de tests statistiques, le problème de la légitimité des tests statistiques en tant que définition adéquate de la notion de SNA n'est pas résolu. Premièrement, pourquoi choisir la définition de Martin-Löf plutôt que celles de von Mises, de Chaitin-Kolmogorov, ou de Schnorr ? Deuxièmement, comment être sûr que la définition qu'en donne Martin-Löf est adéquate alors qu'il est impossible d'exhiber ne serait-ce qu'une SNA ? Je réponds ci-dessous à ces deux questions.

3.1.5 La thèse de Martin-Löf-Chaitin

Pourquoi la notion intuitive de SNA est correctement formalisée par la définition de Martin-Löf plutôt que par celles de Chaitin-Kolmogorov ou de Schnorr ? En fait, aucune définition n'est plus adéquate qu'une autre puisqu'elles sont toutes les trois équivalentes [Chaitin, 1977]. En d'autres termes :

Définition (Définition d'une SNA)
Soit un nombre réel $r \in [0, 1]$. Ce nombre est aléatoire si c'est une suite de Martin-Löf-Chaitin (suite MLC), à savoir si et seulement si :

1. *Il surmonte tous les tests statistiques effectifs - tests de Martin-Löf.*
2. *Il est incompressible - incompressibilité de Chaitin.*
3. *Il n'existe pas de martingale gagnante - martingale de Schnorr.*

Cette équivalence des formalisations a donné lieu à la formulation d'une thèse nommée ou :

Définition (Thèse de Martin-Löf-Chaitin)
Les suites de nombres aléatoires sont les suites de Martin-Löf-Chaitin. En termes concis, les SNA sont les suites MLC.

La thèse MLC illustre le consensus qui existe au sein de la communauté scientifique au sujet de la définition d'une SNA. Néanmoins, sur quels arguments ce consensus est-il fondé ? Autrement dit, comment être sûr que la notion de MLC est la bonne définition d'une suite de nombres aléatoires ?

La stratégie mise en œuvre pour justifier la thèse MLC ressemble fortement à celle qui est utilisée pour justifier la thèse de Church-Turing (TCT). Plus particulièrement, cette stratégie consiste à fournir un ensemble d'arguments considérés comme suffisants pour que l'on s'accorde sur la vérité de la thèse. En fait, la similarité entre les stratégies de justification de la thèse MLC et de la TCT va plus loin que la simple analogie car les arguments en faveur de la TCT - arguments que l'on juge suffisants pour considérer cette thèse comme vraie - peuvent être traduits en arguments en faveur de la thèse MLC. Ainsi, puisque l'on considère la TCT comme étant vraie à partir de ces arguments, on doit considérer la thèse MLC comme étant vraie à partir de ces mêmes arguments.

La comparaison des arguments en faveur de la TCT et de la thèse MLC a été menée à bien par Delahaye [Delahaye, 1993] [Delahaye, 2011]. Voici un résumé des principaux arguments en faveur de la TCT pouvant être traduits en arguments en faveur de la thèse MLC.

Adéquation avec l'intuition au moyen d'exemples. Pour la TCT, l'adéquation avec l'intuition consiste à montrer à partir d'exemples que les fonctions que l'on considère généralement comme calculables sont calculables par MT (Turing-calculables). Inversement, pour la thèse MLC, l'adéquation avec l'intuition consiste à établir à partir d'exemples que les suites de nombres que l'on considère comme n'étant pas aléatoires ne sont pas des suites MLC.

TCT Les fonctions usuelles telle que $f : n \to 2n$, $f : n \to n!$ et $f : n \to n$-ième nombre premier sont calculables au sens intuitif du terme et elles sont Turing-calculables. La TCT n'est donc pas trop restrictive.

Thèse MLC Les suites de nombres telles que $(00000\ldots)$ et (0101010) ne sont pas aléatoires au sens intuitif du terme et elles ne sont pas des suites MLC. La thèse MLC n'est donc pas trop tolérante.

Adéquation avec l'intuition au moyen de contre-exemples. Cet argument est l'argument complémentaire du précédent. Pour la TCT, il consiste à montrer qu'il existe des fonctions non Turing-calculables et qu'il n'existe aucune raison de penser qu'elles puissent être intuitivement calculables. Inversement pour la thèse MLC, il consiste à établir qu'il existe des suites MLC et qu'il n'existe aucune raison de penser qu'elles puissent ne pas être aléatoires.

TCT A partir de la technique de diagonalisation, on peut obtenir des fonctions non Turing-calculables telles que la fonction arrêt des MT pour lesquelles aucun argument en faveur de leur calculabilité effective n'a été fourni. La TCT n'est donc pas trop tolérante. .

Thèse MLC A partir d'arguments non constructifs provenant de la théorie de la mesure et en utilisant des arguments plus directs *via* la définition

du nombre *Omega de Chaitin*[22], il est possible de montrer qu'il existe des suites MLC. La thèse MLC n'est donc pas trop tolérante.

Equivalence des formalisations. Comme je l'ai écrit au chapitre 1, l'argument le plus sérieux en faveur de la TCT est celui de l'équivalence des formalisations. Cet argument consiste à dire que la notion de procédure effective est correctement formalisée par celle de MT, puisque toutes les formalisations qui ont été proposées se sont révélées être équivalentes à la MT. Similairement pour la thèse MLC, l'argument est de dire que la notion de SNA est correctement formalisée par celle de suite MLC puisque l'ensemble des formalisations de SNA lui sont équivalentes.

Toutefois, l'équivalence des formalisations est un argument qui n'a pas la même force lorsqu'on se place du côté de la TCT et lorsqu'on se place du côté de la thèse MLC [Delahaye, 2011, p. 130]. En effet, il existe à ce jour des centaines de formalisations de la notion de procédure effective tandis qu'il n'en existe qu'une poignée pour la notion de SNA. De plus, l'équivalence des formalisations n'est pas un argument suffisant pour démontrer qu'une formalisation correspond adéquatement à une notion intuitive. A titre d'exemple, la notion de fonction primitive récursive ne constitue pas une formalisation adéquate de la notion de procédure effective malgré la myriade de définitions qui lui sont équivalentes[23].

Argument de robustesse. L'argument de robustesse consiste à montrer qu'une formalisation reste cohérente même si certaines modifications sont effectuées.

TCT On peut montrer que certaines modifications opérées sur la MT ne modifient pas la classe des fonctions Turing-calculables. En particulier,

22. En théorie algorithmique de l'information, le nombre Oméga de Chaitin est un nombre réel défini comme étant la probabilité qu'un programme informatique auto-délimité - dont chaque bit est généré aléatoirement - finira par s'arrêter [Chaitin, 1987].

23. Pour rappel, la première formalisation de la notion de procédure effective a été celle des fonctions primitives récursives. Toutefois, la découverte de la fonction d'Ackermann - fonction calculable par procédure effective mais qui n'est pas primitive récursive - a définitivement rejeté l'équivalence entre les fonctions effectivement calculables et les fonctions primitives récursives.

augmenter le nombre de ses rubans, la transformer en machine de Turing non déterministe ou faire travailler plusieurs MT en parallèle ne permet pas de calculer des fonctions non Turing-calculables.

Thèse MLC La notion de SNA résiste aussi à certaines modifications au sein de la formulation de ses définitions. La plus remarquable de ces modifications provient de la formalisation de Chaitin qui utilise la notion de complexité algorithmique K d'une suite de nombre x de longueur n - notée $C(x_n)$. Plus exactement, si c est une constante, les notions de SNA définies à partir des conditions $C(x_n) > n - c$ ou $C(x_n) - n$ sont équivalentes lorsque n tend vers l'infini.

Argument de résistance à des propositions concurrentes. Cet argument ne fait pas partie de ceux que j'avais choisis pour justifier la TCT au chapitre 1, néanmoins, il met en lumière une différence notable entre la TCT et la thèse MLC du point de vue de leur justification.

TCT Du point de vue de la TCT, l'argument de résistance à des propositions concurrentes possède une réelle force. Depuis plus de 50 ans, aucune thèse concurrente ne permet d'exprimer ne serait-ce que certains doutes à l'encontre de la TCT. Au chapitre 1, j'ai par exemple expliqué que l'argument de Bowie selon lequel il existe des fonctions récursives qui ne sont pas calculables est fallacieux. De manière similaire, j'ai montré dans ce même chapitre que la thèse selon laquelle l'hyper-calcul peut remettre en cause la TCT est fausse puisque les notions d'hyper-calcul et de calcul effectif ne sont pas en opposition.

Thèse MLC Du point de vue de la thèse MLC, l'argument est plus fragile. En effet, même si la grande majorité des spécialistes du domaine soutiennent la thèse MLC [24], certaines théories concurrentes - et en particulier une définition soumise par Schnorr - ne sont pas encore éliminées à ce jour [Schnorr, 1977]. D'après von Lambalgen, la découverte de nouveaux résultats mathématiques pouvant entraîner certaines évolutions au sein de la théorie des suites aléatoires n'est pas impossible [von

24. Citons par exemple Kolmogorov et Upensky [Kolmogorov and Uspensky, 1987], Gacs [Gacs, 1986] et Levin [Levin, 1984] parmi les principaux spécialistes.

Lambalgen, 1987]. Van Lambagen défend sa thèse en expliquant que la théorie actuelle proposée par Martin-Löf n'est pas constructive car elle suppose l'axiome du choix, mais qu'une nouvelle théorie pourrait être créée dans un univers ensembliste dénué de cet axiome.

Les arguments qui viennent d'être présentés devraient être suffisants pour considérer la thèse MLC comme vraie. En effet, si l'on accepte les deux prémisses ci-dessous alors il est nécessaire d'accepter la conclusion selon laquelle la thèse MLC est vraie :

Prémisse 1 Les arguments invoqués afin de justifier la TCT et la thèse MLC sont identiques d'un point de vue structurel.

Prémisse 2 Les arguments invoqués afin de justifier la TCT sont suffisants pour la considérer comme vraie.

Conclusion Les arguments invoqués afin de justifier la thèse MLC sont suffisants pour la considérer comme vraie.

De cette conclusion suit que la notion de SNA est correctement définie par les approches de Martin-Löf ou de Chaitin.

Ce dernier résultat dissout de fait une des difficultés énoncées à l'encontre des HM utilisant l'aléatoire comme source d'hyper-calcul. Pour rappel, ces difficultés sont (1) de définir ce qu'est une suite de nombres aléatoires ; (2) de produire une telle suite ; et (3) de vérifier qu'elle est bien aléatoire. Comme je l'ai dit en introduction, la troisième difficulté ne sera pas étudiée dans ce chapitre mais dans le suivant.

Bien que la première difficulté soit levée, cela n'est pas suffisant pour considérer les stratégies utilisant l'aléatoire comme source d'hyper-calcul comme étant plus prometteuses que celles fondées sur la théorie de la relativité générale par exemple. Il me reste en particulier à expliquer de quelle manière ces HM peuvent produire une SNA et surtout pourquoi la production d'une telle suite permet d'hyper-calculer. Ces explications font l'objet de la fin de ce chapitre.

3.2 Hyper-machines à oracle aléatoire

Avant de présenter en détails les propositions de Stannett et de Calude fondées sur la notion d'oracle aléatoire, je vais tout d'abord expliquer comment fonctionne une HM à oracle.

3.2.1 Fonctionnement d'une hyper-machine à oracle

L'idée de calculer avec l'aide d'un oracle provient du travail de Turing [Turing, 1939]. Turing a en effet conçu un type particulier de machines nommées *machines de Turing à oracle* (OM). Les OM sont des MT composées d'un élément supplémentaire appelé *oracle*. Plus précisément, un oracle est une *boîte noire* qui possède un fonctionnement interne non spécifié et qui a la capacité de fournir les résultats de fonctions non Turing-calculables :

> Supposons que l'on nous fournisse des moyens non spécifiés capables de résoudre des problèmes de théorie des nombres ; une sorte d'oracle. Nous n'irons pas plus loin concernant la nature de cet oracle si ce n'est que ce ne peut pas être une machine. A l'aide de l'oracle nous pouvons concevoir un nouveau type de machines (appelons-les *o*-machines), dont l'un des processus fondamentaux est de résoudre un problème de théorie des nombres donné [Turing, 1939, p.167].

Même si l'OM est une HM, son fonctionnement n'est pas assez précis pour comprendre comment elle peut hyper-calculer. C'est pourquoi Copeland et Proudfoot ont proposé une autre définition de l'OM [Copeland and Proudfoot, 1999, p. 103]. De leur point de vue, une OM est une MT composée de deux éléments : un dispositif capable d'exécuter des mesures d'une précision arbitraire et un espace mémoire qui contient une valeur précise appelée τ. Le nombre τ est un nombre représenté par une suite infinie de 0 et de 1 représentant les valeurs d'une fonction non Turing-calculable [25]. Si cette

25. Les résultats d'une fonction de $\mathbb{N}$ dans $\mathbb{N}$ peuvent en effet être représentés par un nombre réel. La première définition que donne Turing du concept de calculable concerne justement les nombres calculables : « D'après ma définition, un nombre est calculable si ses décimales peuvent être inscrites par une machine » [Turing, 1936, p. 116]. Ainsi un nombre est calculable si une MT peut énumérer une à une ses décimales.

fonction est notée d, le $n^{\text{ème}}$ symbole de τ représente $d(n)$ à savoir 0 ou 1. Et si l'on souhaite avoir accès au résultat $d(239208)$, le dispositif mesure le 239208-ième symbole de τ et en fournit la valeur. Par conséquent une OM ayant en mémoire un nombre dont les décimales codent les résultats d'une fonction non Turing-calculable peut calculer davantage de fonctions que la MT.

L'apport de Copeland et de Proudfoot est très important car il permet de définir précisément ce qu'est un oracle. Un oracle est ainsi une suite infini de nombres correspondant aux valeurs d'une fonction non Turing-calculable ; la question étant évidemment de montrer comment produire un tel nombre. C'est ici que la notion de SNA rentre en jeu.

3.2.2 Nombres non Turing-calculables et nombres aléatoires

Ici encore, la thèse selon laquelle une machine qui serait équippée d'un élément aléatoire pourrait faire plus qu'une MT provient des travaux de Turing [Turing, 1950]. Il est cependant fondamental de distinguer deux types d'aléatoire sans quoi tout ordinateur actuel fournissant un nombre aléatoire *via* l'instruction basique **RAND** [26] serait une HM. Plus précisément, on doit faire la différence entre les *processus aléatoires* et les *processus pseudo-aléatoires*. Un processus aléatoire est un processus qui produit une suite de nombres aléatoires similaire aux SNA, c'est-à-dire une processus qui produit une suite MLC. Un processus pseudo-aléatoire produit en revanche une suite de nombres qui *apparait* comme étant aléatoire mais qui est en fait produite par un algorithme.

Ces générateurs de suites pseudo-aléatoires appelés *algorithmes pseudo-aléatoires* sont fréquemment utilisés pour deux raisons. D'une part, créer une SNA qui possède toutes les caractéristiques de l'aléatoire est impossible. En effet, d'après la définition de Martin-Löf, il est nécessaire d'effectuer une infinité de tests statistiques afin de vérifier que cette suite est bien aléatoire [Martin-Löf, 1966]. Pour reprendre la citation de von Neumann : « quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires

26. L'instruction **RAND** est issue du langage de programmation C ; elle a pour fonction d'attribuer *pseudo-aléatoirement* un nombre à une ou plusieurs variables.

est, bien sûr, en train de commettre un péché » [von Neumann, 1951]. D'autre part, l'utilisation d'algorithmes permet d'implémenter sans difficulté de tels générateurs pseudo-aléatoires dans des ordinateurs qui sont ensuite utilisés en informatique, en cryptographie et dans les jeux de hasard.

La plupart des algorithmes pseudo-aléatoires tentent de produire des données en sortie qui sont uniformément distribuées. Il existe toutefois des différences d'efficacité entre ces algorithmes qui dépendent de leur plus ou moins bonne capacité à produire des SNA. Par exemple, un des algorithmes les plus connus - appelé *méthode du carré médian* - possède un défaut majeur lié à sa période qui est très courte [von Neumann, 1951]. Cette méthode consiste à choisir un nombre (la graine), à l'élever au carré et à extraire les chiffres située au milieu du résultat - la donnée en sortie du générateur. Ces chiffres sont ensuite réutilisés comme nouvelle graine. Voici un exemple d'une suite d'instructions qui permet d'appliquer la méthode du carré médian :

1. Choisir la graine, disons 1111.
2. Elever au carré la graine, soit $1111^2 = 1234321$.
3. Extraire les chiffres du milieu, soit 3432. Ce résultat devient la nouvelle graine.
4. Recommencer l'étape 2 et ainsi de suite.

Le principal défaut de la méthode du carré médian est sa courte période ayant pour origine la graine dont dépend la qualité des données en sortie. En effet, si la graine est une suite uniquement constituée d'un ensemble de 0 alors l'algorithme produit toujours des 0. Une telle graine est considérée comme *un état absorbant* qui implique la fin de la période de l'algorithme. L'efficacité de la méthode du carré médian est donc faible car l'expérience montre qu'un tel état absorbant est très vite atteint lorsque l'algorithme est utilisé.

Même si de nombreux autres algorithmes pseudo-aléatoires bien plus efficaces que la méthode du carré-médian ont été conçus [Knuth, 1998], ils ne sont d'aucune utilité pour une HM qui possèderait un tel algorithme en guise d'oracle. Il est en effet prouvé que les ordinateurs actuels ou les MT utilisant des processus pseudo-aléatoires calculent exactement les fonctions Turing-

calculables [De Leeuw et al., 1956]. En revanche, si une machine théorique ou physiquement construite est capable de produire une suite de nombres aléatoires - par opposition à pseudo-aléatoires - elle serait capable de fournir les valeurs d'une fonction non Turing-calculable.

Un raisonnement très simple fondé sur la cardinalité permet de montrer que la suite aléatoire produite par la machine serait non Turing-calculable avec une probabilité égale à 1 [Calude and Svozil, 2008]. Il existe en effet un nombre indénombrable de suites infinies de 0 et de 1 tandis qu'il n'existe qu'un nombre dénombrable de suites infinies de 0 et de 1 qui sont Turing-calculables. Ainsi, en supposant que chaque suite possède la même probabilité d'être produite par le processus aléatoire, la probabilité qu'un processus aléatoire génère une suite Turing-calculable est égale à 0. Il suit que la suite produite est non Turing-calculable avec une probabilité égale à 1.

Les propositions de Stannett et de Calude ont en commun la thèse selon laquelle une machine qui produit une SNA est une HM car la probabilité que la suite produite soit non Turing-calculable est égale à 1. Cependant, la différence entre ces deux propositions réside dans l'origine du processus aléatoire. Expliquons à présent en détails ces deux propositions.

3.2.3 Les propositions de Stannett et de Calude

Les HM que proposent Stannett et Calude peuvent être considérées comme des HM à oracle car elles utilisent de l'information non Turing-calculable issue de la nature. Cette information est d'une part non Turing-calculable car elle prend la forme d'une SNA dont les nombres représentent les valeurs d'une fonction non Turing-calculable. Cette information est d'autre part *issue de la nature* au sens où c'est un processus physique qui produit la SNA : un processus quantique [Calude, 2005] ou un processus radioactif [Stannett, 2003]. Je commence par présenter ces deux propositions, puis j'explique pourquoi elles sont selon moi plus prometteuses que toutes celles que j'ai présentées jusqu'à présent.

La stratégie de Stannett. La stratégie que propose Stannett pour produire une SNA est fondée sur l'utilisation d'un échantillon radioactif [Stan-

nett, 2003]. Plus particulièrement, cet échantillon se désintègre *via* la *désintégration* α, processus pouvant être considéré comme une forme de fission nucléaire où le noyau père d'un atome se scinde en deux noyaux fils dont l'un est un noyau d'hélium. En conséquence, la masse de l'échantillon diminue au cours des désintégrations α successives.

Afin de produire un nombre aléatoire, il est d'une part nécessaire de déterminer la masse totale pouvant être perdue lors de la désintégration et d'autre part la masse minimale pouvant être détectée par les appareils de mesure qui sont à disposition. Ensuite, un échantillon suffisament important est collecté pour veiller à ce que la masse perdue lors de la désintégration soit aisément détectée par les appareils de mesure. Une *valeur seuil* est alors choisie entre la masse minimale pouvant être détectée par les appareils et la masse totale pouvant être perdue lors de la désintégration, et un chronomètre est activé. Enfin, le nombre aléatoire produit par le système est égal au nombre de secondes qui s'écoule avant que la masse correspondante à la valeur seuil choisie ne soit perdue.

La thèse selon laquelle ce nombre est aléatoire est soutenue par l'hypothèse stantard de la théorie de la radioactivité, hypothèse considérée comme valide pour tout échantillon radioactif. Cette hypothèse est constituée de deux points importants :

1. La désintégration α est formalisée par une loi statistique qui décrit un processus aléatoire.
2. Après un certain intervalle de temps t, appelé *temps de demi-vie*, la moitié des noyaux radioactifs d'un échantillon est désintégrée [27].

A partir de ces deux points, il est aisé d'expliquer pourquoi le résultat de l'expérience présentée plus haut est un nombre aléatoire. Premièrement, puisque la valeur seuil choisie est strictement inférieure à la masse totale de l'échantillon, cette valeur devra être atteinte après un nombre fini de demi-vies. Deuxièmement, puisque la désintégration est supposée être aléatoire, le nombre de secondes qui s'écoulent avant que la valeur seuil soit atteinte

27. Il est important de noter que deux demi-vies ne correspondent pas à la désintégration de la totalité des noyaux radioactifs de l'échantillon car la perte de matière supplémentaire ne porte plus que sur la moitié restante et non sur le total initial.

doit aussi être considéré comme aléatoire. En d'autres termes, le système implémente un véritable générateur de nombres aléatoires.

La stratégie de Calude. La stratégie de Calude consiste à fixer sur un ordinateur un appareil capable de produire une SNA *via* un processus quantique [Calude, 2005]. Un tel appareil a été conçu par l'entreprise *ID Quantique* et se nomme *Quantis*[28]. Quantis exploite un processus quantique élémentaire qui permet de produire une suite de nombres aléatoires [Jennewein et al., 2000]. La lumière est en effet composée de particules élémentaires appelées *photons* qui, dans certaines situations, affichent un comportement aléatoire. La projection de ces derniers sur un miroir semi-transparent en est un exemple. Plus précisément, un photon qui est projetée vers un miroir semi-transparent a 50% de chance d'être réfléchi par le miroir et 50% de chance de le traverser[29]. En associant à ces deux évènements - réflection, transmision - les nombres 0 et 1, il est possible de traduire les mesures des positions des photons en une suite aléatoire de 0 et de 1. Un ordinateur équipé de Quantis peut en théorie produire une suite arbitrairement longue de nombres ou *bits quantiques* aléatoires qui ne peuvent pas être produits par une MT. Quantis est donc vu comme un oracle qui fournit de l'information non calculable issue de la nature - *via* un processus quantique - dans le but de surpasser les capacités de la MT.

Pourquoi ces stratégies sont les plus prometteuses. Même si je souhaite défendre que les propositions de Calude et de Stannett sont celles qui ont le plus de chances de voir le jour, ces deux propositions ne sont pas exemptes de critiques. Ces critiques sont d'après moi au nombre de deux, l'une étant théorique et l'autre pratique.

La première critique est d'ordre théorique et consiste à dire qu'il n'est pas certain que les suites de nombres fournies par les HM de Stannett et de Calude soient aléatoires. En effet, ces suites sont supposées être aléatoires d'après

28. `http://www.idquantique.com/`.

29. Ce résultat provient du postulat de Born : lorsqu'un système quantique clos qui est dans l'état $V = (v_{1,1}, v_{2,1}, \ldots, v_{n,1}^T)$ est mesuré, la mesure conduit au résultat i avec une probabilité égale à $|v_{i,1}|^2$.

certaines hypothèses physiques : l'hypothèse statistique de désintégration radioactive et le postulat de Born. Ces hypothèses - conditions *sine qua non* pour que l'oracle produise de l'aléatoire - sont par définition des énoncés non démontrés. Ainsi, ce n'est qu'en admettant ces hypothèses - et non en les démontrant - qu'il est possible d'affirmer que les suites fournies par les machines de Stannett et de Calude sont aléatoires, autrement dit que ces machines sont bien des HM.

La seconde critique - qui est de loin la plus décisive - est quant à elle d'ordre pratique. En supposant que les suites fournies par les HM de Calude et de Stannett soient réellement aléatoires, ces suites représenteraient les valeurs de fonctions non Turing-calculables *uniquement* dans le cas où elles seraient infinies. En effet, toute suite finie est par définition Turing-calculable car il suffit de programmer une MT qui reproduise un à un les nombres qui composent cette suite. Or la production d'une suite infinie est pour l'instant irréaliste car il faudrait par exemple disposer - dans le cas de l'HM de Calude - d'une source d'énergie inépuisable qui projète *ad infinitum* des photons vers un miroir semi-transparant.

Malgré ces faiblesses, je pense néanmoins que la stratégie de construction d'une machine à oracle aléatoire () est la plus prometteuse en vue de démontrer la TPHC. Autrement dit, je soutiens que la construction physique des HM introduites par Stannett et Calude est plus réaliste que la construction des HM newtoniennes, relativistes et adiabatiques. Mon argumentation se fonde sur deux des principaux points forts que possèdent les OMA :

1. **Le dispositif nécessaire pour faire fonctionner l'OMA est actuellement construit.** Ce dispositif est en effet essentiellement composé de deux parties pouvant être construites : un ordinateur - tel que ceux construits actuellement - et un générateur de nombres aléatoires - tel que Quantis. La situation est cependant loin d'être la même pour les trois propositions concurrentes. Pour l'HM newtonienne, aucun ordinateur pouvant se dupliquer en une version miniaturisée n'a été constuit. Pour l'HM relativiste, on ne dispose d'aucune technologie qui permettrait de propulser le dispositif - composé de deux ordinateurs - vers l'horizon d'un trou noir. Pour le dispositif de l'HM adiabatique, aucune

implémentation physique d'opérateurs hamiltoniens ayant un nombre infini de niveaux d'énergie n'a été réalisée avec succès.

2. **Les hypothèses sur lesquelles les OMA sont fondées - à savoir le postulat de Born et l'hypothèse statistique de désintégration radioactive - sont confirmées par l'expérience**. Cela signifie que les expériences physiques qui sont effectuées en continu par les scientifiques sont en accord avec les hypothèses de la théorie ; l'expérience d'Aspect en est un célèbre exemple [30]. On ne peut en revanche pas en dire autant des autres propositions. Même si la proposition de Kieu se fonde sur le thèorème adiabatique de la mécanique quantique, aucune expérience n'a pu confirmer son utilité pour résoudre des problèmes indécidables. Dans le cas de la proposition relativiste, aucune expérience n'a confirmé l'existence - au sens physique du terme - des espace-temps de Malament-Hogarth, clé de voûte de l'HM à trous noirs. Enfin, la proposition newtonienne est même infirmée par l'expérience puisqu'elle montre que la théorie newtonienne n'est pas adéquate à l'échelle microscopique.

Si la construction physique des propositions de Calude et de Stannett est très incertaine, ces deux arguments sont d'après moi suffisants pour montrer qu'elles sont plus prometteuses que les tentatives issues des théories newtonienne, relativiste et quantique. Elles sont plus prometteuses au sens précis où leur possible construction physique nécessite peu de présupposés théoriques et est assez réaliste d'un point de vue pratique. Les recherches entreprises afin de démontrer la TPCH devraient donc se concentrer en priorité sur les OMA.

4 Conclusion

Dans ce chapitre, j'ai tenté de défendre une thèse qui ne fait pas l'unanimité dans la communauté scientifique. Cette thèse consiste à dire que cer-

30. Cette expérience est la première expérience qui a réfuté de manière satisfaisante les inégalités de Bell dans le cadre de la physique quantique, validant ainsi le phénomène d'intrication quantique et falsifiant l'hypothèse d'une théorie quantique déterministe.

taines propositions introduites en vue de construire physiquement une HM ont des chances d'aboutir. Plus précisément, j'ai essayé de montrer que les propositions de Stannett [Stannett, 2003] et de Calude [Calude, 2005] fondées sur la notion d'aléatoire sont plus prometteuses que celles de Davies [Davies, 2001], Shagrir & Pitowsky [Shagrir and Pitowsky, 2003] et Kieu [Kieu, 2001] issues des théories newtonienne, relativiste et quantique.

J'ai commencé par montrer dans les deux premières sections que je suis en accord avec la conclusion de la communauté scientifique selon laquelle et il fort peu probable que les HM de Davies, de Shagrir & Pitowsky et de Kieu soient un jour construites. Voici un résumé des obstacles inhérents à ces propositions :

HM newtonienne de Davies Le problème majeur est que la propriété physique essentielle sur laquelle se fonde cette HM, à savoir qu'il n'existe pas de limite physique à la division de l'espace et du temps, n'est pas une propriété que possède le monde réel. D'après la théorie quantique, la matière est en effet atomique à l'échelle microscopique ce qui implique l'existence de limites physiques sur la taille des particules de matière. La physique quantique permet en particulier d'établir que la miniaturisation d'un des composants de l'HM de Davies - son chronomètre - admet une limite à partir de laquelle on peut déduire une taille et un poids minimum que doit posséder un chronomètre pour fonctionner. Cette limite empêche ainsi l'HM de Davies d'être construite puisqu'il est impossible de miniaturiser à volonté un de ces composants sans altérer son fonctionnement.

HM relativiste de Shagrir et Pitowsky Même si la théorie relativiste n'est pas infirmée, cette HM soulève néanmoins trois problèmes. Tout d'abord, il n'existe aucune preuve de l'existence - au sens physique - des espace-temps de Malament-Hogarth censés permettre à l'HM d'exécuter un ST. Ensuite, l'existence de ces espace-temps n'implique pas que l'on dispose un jour de la technologie suffisante pour atteindre le trou noir dont l'horizon est un espace-temps de Malament-Hogarth. Enfin, l'exécution d'une infinité d'étapes en un temps fini autour de ce trou noir serait remise en cause au cas où l'hypothèse de Hawking

selon laquelle tout trou noir disparaît après un intervalle de temps fini s'avérerait vraie.

HM quantique de Kieu Les problèmes physiques et épistémologiques liés à l'HM de Kieu sont très nombreux. Ces problèmes physiques sont liés aux difficultés (1) d'implémenter physiquement certains opérateurs hamiltoniens ayant un nombre infini de niveaux d'énergie ; et (2) d'obtenir physiquement les états fondamentaux. Du côté des problèmes épistémologiques citons en particulier celui de déterminer le moment où le système doit être mesuré. Ce moment est très difficile à déterminer car le fait de laisser le système évoluer - ce qui revient à ne pas mesurer l'état du système - augmente simultanément la probabilité d'obtenir le résultat d'une fonction non Turing-calculable.

J'ai ensuite défendu dans la dernière section de ce chapitre que les HM utilisant l'aléatoire comme source d'hyper-calcul sont plus prometteuses que leurs analogues newtoniennes, relativistes et quantiques. La défense des OMA s'est faite à partir de trois arguments :

1. **Les critiques le plus souvent énoncées à l'encontre de la notion de SNA peuvent être surmontées**. Ces critiques consistent (1) à reprocher à la notion de SNA de n'être pas correctement définie d'un point de vue mathématique ; et (2) à affirmer qu'il est impossible de produire des SNA - par opposition à des suites pseudo-aléatoires. J'ai toutefois expliqué que la notion de SNA peut être correctement formalisée à partir des travaux de Martin-Löf et de Chaitin, et qu'il est possible de produire de l'aléatoire à l'aide de deux processus physiques : la désintégration radioactive et la projection de photons sur un miroir semi-transparent.

2. **Le dispositif nécessaire pour faire fonctionner l'OMA est construit.** Ce dispositif est en effet essentiellement composé de deux parties pouvant être construites : un ordinateur et un générateur de nombres aléatoires. D'une part, le générateur de l'HM de Calude appelé *Quantis* est actuellement construit ; d'autre part, Stannett décrit précisément le protocole expérimental nécessaire à la production de SNA à partir de

la désintégration radioactive.

3. **Les hypothèses sur lesquelles les OMA sont fondées sont confirmées par l'expérience**. Cela signifie que les expériences physiques qui sont effectuées en continu par les scientifiques sont en accord avec le postulat de Born et l'hypothèse statistique de désintégration radioactive sur lesquelles reposent les OMA.

Même si les propositions de Stannett et de Calude sont prometteuses au sens où les problèmes mathématiques et physiques couramment énoncés à leur encontre peuvent être résolus, je souhaite néanmoins soulever un problème épistémologique lié à la vérification des suites de nombres aléatoires.

Ce problème épistémologique que je nomme *problème de la vérification* peut être énoncé de la manière suivante [Franchette, 2012] : supposons qu'une HM est physiquement construite, est-il possible de vérifier que l'HM calcule une fonction non Turing-calculable ? Dans le cas de l'OMA, cela revient à déterminer si la suite produite par l'HM est bien aléatoire. Cependant, tous les tests dont on dispose actuellement sont insuffisants pour différencier une suite aléatoire d'une suite pseudo-aléatoire ; les meilleurs générateurs de nombres pseudo-aléatoires produisant des suites qui sont impossibles à différencier des suites produites par un générateur de nombres aléatoires tel que Quantis. Par conséquent, comment être certain qu'une HM fondée sur l'aléatoire fournit une suite aléatoire et non pas pseudo-aléatoire ?

Le problème de la vérification n'est pas uniquement soulevé par les HM de Stannett et de Calude. En effet, je pense que ce problème concerne toute HM qui est physiquement construite. Plus encore, je pense que le problème de la vérification est un véritable obstacle à une possible démonstration de la TPHC, et ce même si une HM est physiquement construite. Le chapitre suivant est consacré à l'étude de ce problème et à ses conséquences pour l'hyper-calcul.

Chapitre 4

Le problème de la vérification

> The upshot is that we cannot dismiss the possibility that we will someday have reasons for believing that some physical device computes a Turing uncomputable function that are just as good as the reasons that we currently have for believing that our hand calculators compute addition.
>
> Carol Cleland

Les trois chapitres précédents ont eu pour but de défendre la thèse générale suivante : la construction physique d'hyper-machines (HM) - à savoir de dispositifs capables d'hyper-calculer - pourrait remettre en cause une position philosophique sur les ordinateurs défendue depuis la seconde moitié du XX^{e} siècle, position selon laquelle l'étude des ordinateurs n'a en principe aucune influence sur les limites du calcul.

Plus précisément, j'ai tout d'abord montré dans le chapitre 1 que l'hyper-calcul - à savoir le calcul de fonctions non calculables par machine de Turing (MT) - n'est pas logiquement contradictoire et qu'il est compatible avec la thèse de Church-Turing (TCT), clé de voûte de la théorie de la calculabilité.

J'ai ensuite expliqué au chapitre 2 pourquoi la construction physique d'un dispositif capable d'hyper-calculer - autrement dit une HM - aurait pour conséquence de replacer les ordinateurs au centre de l'étude des limites du calcul. J'ai enfin argumenté dans le chapitre 3 en faveur de la thèse physique de l'hyper-calcul (TPHC) - thèse affirmant qu'il est possible de construire physiquement une HM - en défendant les propositions fondées sur l'utilisation de l'aléatoire comme source d'hyper-calcul.

Le présent chapitre tente quant à lui de dépasser les thèses que j'ai défendues dans les trois chapitres précédents afin d'en montrer les limites. Je vais plus précisément soulever un problème épistémologique qui pourrait faire obstacle à la démonstration de la TPHC. Ce problème - que je nomme - est énoncé de la manière suivante :

Définition (Problème de la vérification)
Supposons que l'on dispose d'un dispositif supposé être une HM, peut-on vérifier que le dispositif hyper-calcule ?

Si la réponse à ce problème est négative, la démonstration de la TPHC serait confrontée à une difficulté supplémentaire. En effet, si les obstacles liés à la construction physique d'une HM - qui sont à la fois théoriques et pratiques - arrivaient à être surmontés, il serait néanmoins impossible d'affirmer que le dispositif qui a été construit hyper-calcule. Autrement dit, même si la TPHC est vraie - le dispositif construit est bien une HM - il serait impossible de prouver que l'on a réussi.

Copeland a été l'un des premiers a défendre explicitement que la vérification des valeurs de fonctions non Turing-calculables pouvait être un réel problème pour la TPHC :

> Il y a un problème épistémologique lié à l'hyper-calcul. Supposons que le génie de Laplace dise 'Voici une boîte noire capable de résoudre le problème de l'arrêt' (le problème est soulevé quelle que soit la fonction non calculable par machine de Turing qui est considérée). Tapez n'importe quel entier x et la boîte fournira la valeur de la fonction arrêt $H(x)$ correspondante. Puisqu'il n'y a pas de méthode systématique pour calculer les valeurs de la

> fonction arrêt, vous n'avez pas de moyen de vérifier si la machine produit les réponses correctes ou non. Même simuler la machine de Turing en question ne vous aidera pas en général : quelle que soit la durée durant laquelle vous regarderez la simulation, vous ne pourrez pas inférer que la machine ne s'arrêtera pas à partir du fait qu'elle ne s'est pas encore arrêtée [Copeland, 2002b, pp. 490-491].

D'après Copeland, si on fait l'hypothèse selon laquelle un dispositif est une HM, il est impossible de montrer que cette hypothèse est vraie. La raison en est que la démonstration de cette hypothèse repose sur notre capacité à vérifier que la machine calcule au moins une fonction non Turing-calculable. Or une telle vérification est impossible car on ne détient pas de méthode systématique - de procédure effective ou d'algorithme - permettant, à partir d'une donnée en entrée, de suivre chacune des étapes du calcul et d'arriver en un temps fini au résultat. Ainsi, toute tentative de vérification des propriétés hyper-calculatoires d'une HM est vouée à l'échec.

Si Copeland ne fait pas d'erreur de raisonnement, son analyse est selon moi incomplète. Considérons en particulier le cas des ordinateurs actuels. D'après le raisonnement de Copeland, il est possible d'affirmer que les ordinateurs calculent puisque l'on détient une méthode systématique - le programme de l'ordinateur - qui permet, à partir d'une donnée en entrée, de suivre chacune des étapes d'un calcul et d'arriver en un temps fini à son résultat. Toutefois, cette conclusion est correcte *seulement* en principe, c'est-à-dire si l'on fait abstraction des ressources physiques.

En pratique - lorsque l'on est limité par le temps et l'espace nécessaires aux calculs - une vérification des résultats fournis par les ordinateurs est en effet impossible à effectuer en suivant chacune des étapes du calcul : l'être humain a besoin des ordinateurs pour exécuter les calculs infaisables manuellement - *cf.* chapitre 2. Pourtant, les ordinateurs sont considérés comme des dispositifs capables d'exécuter des calculs. La preuve en est que notre univers technique se présente aujourd'hui comme un univers d'ordinateurs qui sont indispensables au fonctionnement de notre société. Ils sont le fondement de notre économie - système bancaire - de nos moyens de transports aériens

ainsi que de nos moyens de production d'énergies - électrique et nucléaire. On accorde par conséquent une véritable confiance en ce qui concerne le calcul de nos ordinateurs.

Mais quelle est l'origine de cette confiance ? L'origine de cette confiance réside dans l'existence de méthodes - théoriques et expérimentales - pouvant être utilisées afin d'affirmer qu'un dispositif calcule. Ces méthodes peuvent reposer à la fois sur une approche philosophique centrée sur la notion générale de système physique ou sur une approche plus fidèle à la pratique scientifique centrée sur la notion d'ordinateur :

1. La première approche consiste à déterminer certains critères suffisants pour affirmer qu'un système physique calcule. Par exemple, un système physique calcule selon Putnam s'il existe une *relation d'implémentation* - notion qui sera définie par la suite - entre une description computationnelle telle que la MT et le système physique en question [Putnam, 1975].

2. La seconde approche consiste quant à elle à fournir des techniques de vérification fondées sur les parties logicielle - programme - et matérielle - composants électroniques par exemple - propres aux ordinateurs à partir desquelles on peut vérifier qu'une fonction donnée a été calculée.

De part l'efficacité de ces méthodes - on affirme de fait que les ordinateurs calculent - le problème de la vérification est par conséquent résoluble dans le cas des ordinateurs : il est possible de vérifier - dans un sens que je préciserai dans ce chapitre - qu'un dispositif calcule - par opposition à hyper-calcule.

Peut-il toutefois en être de même dans le cas des HM ? Autrement dit, peut-on vérifier qu'un dispositif hyper-calcule ? Dans ce chapitre, mon but est de répondre à ces deux questions afin de déterminer si le problème de la vérification est résoluble. Plus précisément, ma stratégie consiste à évaluer si les deux approches introduites plus haut - centrées respectivement sur les notions de système physique et d'ordinateur - peuvent être appliquées à l'hyper-calcul.

Voici le plan de mon argumentation. Je vais dans une première section étudier les systèmes physiques en général et les moyens qui permettent d'af-

firmer qu'ils calculent. Cependant, je défendrai que ces moyens ne sont pas pertinents pour déterminer si un système physique hyper-calcule. Dans une seconde section, je vais poursuivre une stratégie différente qui consiste à examiner les méthodes et techniques sur lesquelles s'appuient les scientifiques - informaticiens et ingénieurs - pour affirmer qu'un ordinateur calcule une fonction particuière. Cette analyse se conclura par la formulation d'un ensemble de critères qui, dans le cas où il est satisfait par une machine, atteste que cette machine calcule une fonction Turing-calculable donnée. Je tenterai enfin dans une dernière section de déterminer si l'on peut appliquer ces critères aux HM afin d'affirmer qu'une fonction non Turing-calculable est calculée.

1 Le calcul au sein des systèmes physiques

Dans cette section, je propose d'analyser la première approche qui tente de résoudre le *problème de l'implémentation*, à savoir le problème de déterminer certains critères suffisants pour affirmer qu'un système physique - naturel ou artificiel - calcule. Mon but est plus précisément d'évaluer si cette approche fournit un cadre théorique pouvant être utilisé afin d'affirmer qu'un système physique hyper-calcule. Je commence ainsi par présenter le problème de l'implémentation et sa réponse standard, *l'analyse standard du calcul*. J'explique ensuite que cette analyse conduit au *pancomputationnalisme*, thèse selon laquelle tout système physique calcule - qu'il s'aggisse de fonctions Turing-calculables ou non. Pour finir, je montre que les critères proposés par l'analyse standard ne sont pas suffisants pour déterminer si un système physique hyper-calcule.

1.1 L'analyse standard du calcul

L' est une réponse au problème de l'implémentation, à savoir le problème de déterminer les caractéristiques que doit posséder un système physique pour que l'on puisse affirmer qu'il calcule. Plus précisément, dire qu'un système physique calcule revient à affimer qu'il existe une *relation d'implémentation*

d'une certaine espèce entre une description computationnelle et le système ; le problème central étant de définir et de délimiter l'expression *d'une certaine espèce*.

1.1.1 L'analyse de Putnam

L'analyse standard du calcul a pour origine les travaux de Putnam énonçant qu'un système qui est décrit avec précision par une description computationnelle $\mathcal{C}$ est un système implémentant $\mathcal{C}$ [Putnam, 1960], [Putnam, 1967], [Putnam, 1975]. L'idée générale défendue par Putnam est de dire qu'un système physique - ou une machine selon ses termes - calcule s'il existe une relation spécifique entre lui et une machine abstraite telle que la MT.

Putnam décrit cette relation d'implémentation à partir de la table d'instructions des MT. La table d'instructions d'une MT est une grille qui possède autant de colonnes que d'états internes dans lesquels la MT peut se trouver et autant de rangées que de types de symboles qu'elle peut manipuler. Par exemple, le comportement d'une MT particulière peut être entièrement caractérisé à partir de sa table d'instructions ce qui justifie le fait que les MT sont souvent définies en termes de tables d'instructions. A titre d'exemple, voici la table d'instructions d'une MT où q_0, q_1, q_2 sont ses états internes et $\Lambda, *$, I les symboles qu'elle peut manipuler [Lavallée, 2008] :

	q_0	q_1	q_2
Λ		IGq_2	ΛDq_0
$*$	$\Lambda\,!$	$*Dq_1$	$*Gq_2$
I	$\Lambda\ Dq_1$	IDq_1	IGq_2

Putnam définit ensuite la relation d'implémentation entre une machine et une MT de la manière suivante :

> Une "table d'instructions" *décrit* une machine si la machine possède des états internes correspondant aux colonnes de la table, et si elle "obéit" aux instructions de la table comme ceci : lorsqu'elle est en train de lire une cellule sur laquelle un symbole s_1 apparaît et qu'elle est, disons, dans l'état B, elle exécute "l'instruction" ins-

> crite dans la rangée et la colonne appropriées de la table (dans ce cas, la colonne B et le symbole s_1). Toute machine qui est décrite par une table d'instructions d'après la précédente illustration est une machine de Turing [Putnam, 1975, p. 365].

A première vue, fonder la relation d'implémentation sur une correspondance entre une machine physique et une MT semble être adéquat. En effet, les machines physiques telles que les ordinateurs actuels partagent certaines propriétés importantes avec la MT - *cf.* chapitre 2. Parmi ces propriétés figurent en particulier l'universalité - les ordinateurs et la MT calculent exactement les mêmes fonctions mathématiques, à savoir les fonctions Turing-calculables - et la programmabilité - ils peuvent être modifiés pour suivre des instructions différentes de celles qui leur ont été attribuées lors de leur création.

Cependant, la proposition de Putnam revêt deux problèmes majeurs. D'une part, la relation entre une machine physique et une MT que propose Putnam repose sur des notions inexpliquées telles que les notions de cellule, de ruban, de symboles ou de tête de lecture [Piccinini, 2010]. Ces notions sont plus précisément des notions *primitives* - au sens où elles ne sont pas définies - qui font partie intégrante du fonctionnement d'une MT ; son fonctionnement étant souvent illustré à partir d'une représentation graphique de son ruban divisé en cellules et de sa tête de lecture - ici représentée par des parenthèses :

...	Λ	(I)	I	*	I	I	I	Λ	Λ	...

D'autre part, la MT n'est pas la seule description computationnelle possible d'une machine physique [Shepherdson and Strurgis, 1963]. Rappelons en particulier que les UMR (*Unlimited Register Machine*) aussi appelées RAM (*Random Access Memory)* modélisent les unités de calcul d'un ordinateur et possèdent les mêmes propriétés que la MT - universalité, programmabilité, *etc.* Ainsi, pourquoi la MT devrait être condidérée comme le modèle de référence sur lequel se fonde la relation d'implémentation alors qu'il existe d'autres modèles qui lui sont équivalents ?

Ces deux problèmes illustrent en partie les raisons qui ont poussé Putnam à abandonner sa description en termes de cellules, symboles et autres notions inexpliquées, et de les remplacer en faisant appel à la *description physique* du système - par exemple une équation différentielle. Le résultat de cette substitution est ce que Godfrey-Smith nomme le SMA (*Simple Mapping Account*) [Godfrey-Smith, 2009].

1.1.2 Le SMA ()

D'après le , un système physique calcule lorsqu'il existe une relation fonctionnelle entre le système et une description computationnelle qui peut être décrite en termes d'*états de transition*. Plus particulièrement, cette description computationnelle n'est pas la MT mais une description équivalente plus générale appelée ().

Un AEF est un ensemble fini de données en entrée I, de données en sortie O et d'états internes S qui sont reliés par des états de transition, à savoir des applications de la forme $\{S_i, I_j\} \rightarrow \{S_k, O_l\}$. Voici par exemple la représentation d'un AEF à partir de ses états de transition où $S = \{S_1, S_2, S_3\}$, $I = \{I_1, I_2\}$ et $O = \{O_1, O_2, O_3\}$:

$$\begin{array}{ll} (S_1, I_1) \rightarrow (S_2, O_1) & (S_1, I_2) \rightarrow (S_3, O_1) \\ (S_2, I_1) \rightarrow (S_3, O_1) & (S_2, I_2) \rightarrow (S_1, O_2) \\ (S_3, I_1) \rightarrow (S_1, O_2) & (S_3, I_2) \rightarrow (S_1, O_3) \end{array}$$

La relation d'implémentation entre un système physique $\mathcal{S}$ et un AEF - ou plus explicitement la relation énonçant que $\mathcal{S}$ est un AEF - est satisfaite si les deux conditions suivantes sont remplies [Chalmers, 1996] :

1. Il existe une application des états de la description physique du système $\mathcal{S}$ vers les états définis par l'AEF telle que les états de transition entre les états de la description physique *reflètent* les états de transition entre les états computationnels.
2. Pour toute transition des états computationnels de la forme $s_1 \rightarrow s_2$ - spécifiée par l'AEF - si le système est dans l'état physique appliqué à s_1 alors le système entre dans l'état physique appliqué à s_2.

Cette formulation peut sembler incomplète puisque les descriptions physiques couramment utilisées - telles que les systèmes d'équations différentielles - assignent la plupart du temps un nombre indénombrable d'états aux systèmes physiques, tandis que les descriptions computationnelles - telles que les tables de transition - assignent généralement un nombre dénombrable d'états. En particulier, si l'on recquiert que l'application de l'ensemble des états computationnels vers l'ensemble des états de la description physique soit bijective, la première condition ne pourra pas être satisfaite pour les systèmes physiques décrits par des équations différentielles.

Pour résoudre ce problème, deux solutions existent. La première est d'inverser la direction de l'application en requérant une application des états computationnels vers un sous-ensemble des états physiques. La seconde qui est la solution la plus utilisée consiste à sélectionner un sous-ensemble des états physiques ou bien des classes d'équivalence des états physiques et de les appliquer vers les états computationnels. Dans ces deux cas, la condition 1 du SMA est remplacée par la condition 1' suivante : il existe une application d'un sous-ensemble des - ou des classes d'équivalence des - états assignés à $\mathcal{S}$ par une description physique définie par la description computationnelle $\mathcal{C}$.

D'une manière générale, l'analyse standard du calcul est très libérale car les applications entre une description computationnelle et les états physiques d'un système sont relativement aisées à construire. Par exemple, voici de quelle manière l'analyse standard attribue à un rocher le calcul d'une porte NOT*.

Considérons que l'on est au petit matin et qu'un rocher est posé sous le soleil. Durant un intervalle de temps, la température du rocher augmente. Plus précisément, la température du rocher évolue de la température T vers les températures T+1 et T+2. Maintenant, considérons une porte NOT* qui applique deux fois une porte NOT dont le comportement calculatoire sur un alphabet binaire est NOT(0) = 1 et NOT(1) = 0. En appliquant la porte NOT* à l'entier 0, on a NOT*(0) = NOT(0) = NOT(NOT(0)) = 0 - à savoir la suite 010. On applique enfin les états physiques T et T+2 à 0, et T+1 à 1 et on conclu d'après l'analyse standard que le rocher implémente une porte NOT*.

La facilité avec laquelle est attribuée l'exécution d'un calcul à un système physique illustre la grande généralité de l'analyse standard. Cette généralité a en particulier permis à certains auteurs de défendre ce que l'on appelle aujourd'hui le *pancomputationnalisme*, thèse selon laquelle tous les systèmes physiques peuvent implémenter tous les calculs pouvant être exécutés par une description computationnelle telle que la MT.

1.2 Le pancomputationnalisme

Quels sont les systèmes physiques capables d'exécuter des calculs ? D'après le pancomputationnalisme, tous les systèmes physiques en sont capables qu'il s'agisse des rochers, des tempêtes ou des moulins hydrauliques. Il existe néanmoins deux formes de pancomputationnalisme :

Définition (Pancomputationnalisme faible et fort)

- *Le affirme que tous les systèmes physiques sont capables d'exécuter certains - par opposition à tous - calculs pouvant être reproduits par un automate à états finis [Chalmers, 1996], [Scheutz, 1999].*
- *Le affirme que tous les systèmes physiques sont capables d'exécuter tous les calculs pouvant être reproduits par un automate à états finis [Putnam, 1988], [Searle, 1992].*

Ces deux formes de pancomputationnalisme ont une base commune, à savoir l'attribution de calculs à tous les systèmes physiques. Elles diffèrent toutefois sur le degré de cette attribution, c'est-à-dire sur la classe de fonctions que les systèmes physiques peuvent calculer. Pour le pancomputationnalisme faible, cette classe ne comprend pas forcément toutes les fonctions Turing-calculables ; tandis que pour le pancomputationnalisme fort, tous les systèmes physiques peuvent calculer l'ensemble des fonctions Turing-calculables.

Il est en outre important de noter qu'aucune de ces deux formes n'affirme que les systèmes physiques calculent *au plus* l'ensemble des fonctions Turing-calculables, elles expliquent uniquement que certains systèmes - pour la forme faible - ou tous les systèmes - pour la forme forte - calculent l'ensemble des fonctions Turing-calculables. A supposer que le pancomputationnalisme soit vrai, cela ne remettrait donc pas en cause l'existence de systèmes physiques

pouvant hyper-calculer puisqu'une HM calcule par définition les fonctions Turing-calculables.

D'après Cleland, un des premiers arguments en faveur du pancomputationnalisme faible a été proposé par Hinckfuss lors d'une discussion en 1978 [Cleland, 2002]. Cet argument - nommé aujourd'hui *l'argument du seau de Hinckfuss* - consiste à dire qu'un seau d'eau reposant sous le soleil est d'une complexité si grande qu'il pourrait, à partir d'une caractérisation adéquate de ses états, être considéré comme implémentant les calculs pouvant être éxécutés par le cerveau humain [1]. L'argument a ensuite été repris par Lycan en considérant un seau d'eau contenant une immense quantité de processus microscopiques :

> Maintenant, est-ce que toute cette activité [...] ne pourrait pas, simplement par chance, réaliser un programme humain ne serait-ce que pendant une brève période (étant données des corrélations adéquates entre certains micro-évènements et les données en entrée, les données en sortie, et les états du programme) ? [Lycan, 1981].

Même si Lycan est d'un point de vue historique un des premiers à défendre le pancomputationnalisme faible, son argument échoue par sa forme - il est informel - et par son fond - il n'est pas généralisé à tous les systèmes physiques. Le premier argument rigoureux en faveur du pancomputationnalisme - à la fois formel et généralisé à l'ensemble des systèmes physiques - a été une fois de plus proposé par Putnam [Putnam, 1988].

1. Le pancomputationnalisme, qu'il soit fort ou faible, est souvent considéré comme une menace pour la théorie computationnelle de la cognition car il rendrait ses arguments vides de sens. Schématiquement, la théorie computationnelle de la cognition a pour but d'expliquer que le cerveau humain un est système physique spécifique qui diffère des autres systèmes physiques. En particulier, cette théorie défend (1) que le cerveau est un système qui calcule ; et (2) que ces calculs peuvent être décrits par un modèle équivalent à la MT. La théorie computationnelle de la cognition suppose ainsi que tous les systèmes physiques ne calculent pas et que parmi les systèmes physiques qui calculent, certains ne calculent pas autant de fonctions que le cerveau humain [Horst, 2003]. Cette théorie deviendrait par conséquent triviale sous l'effet du pancomputationnalisme car le cerveau - comme tout autre système - pourrait être considéré comme une implémentation d'un modèle équivalent à la MT.

1.2.1 Arguments en faveur du pancomputationnalisme

Dans son ouvrage de 1988, Putnam défend le pancomputationnalisme fort en argumentant en faveur de la thèse suivante : tous les systèmes ouverts ordinaires implémentent n'importe quel automate abstrait fini - sans données en entrée ni données en sortie. Pour défendre sa thèse, Putnam suppose que les champs électromagnétique et gravitationnel sont continus et considère (1) un nombre arbitraire d'automates finis dont les tables d'instructions conduisent à la suite d'états $ABABABA$; et (2) un système physique $\mathcal{S}$ arbitraire qui évolue sur un intervalle de temps arbitrairement choisi allant de 12 : 00 à 12 : 07. Pour finir, Putnam explique que $\mathcal{S}$ implémente la suite $ABABABA$ et indique que son argumentation se généralise à tout automate et tout système physique puisque ces derniers ont été choisis de façon arbitraire. Voici le cœur de cette argumentation :

> Soient $t_1, t_2, \ldots t_n$ les débuts des intervalles durant lesquels S est dans une de ses étapes A ou B (dans l'exemple donné, $n = 7$, et les temps en question sont $t_1 = 12 : 00$, $t_2 = 12 : 01$, $t_3 = 12 : 02$, $t_4 = 12 : 03$, $t_5 = 12 : 04$, $t_6 = 12 : 05$, $t_7 = 12 : 06$). Nous appellerons t_{n+1} la fin de l'intervalle en temps réel durant laquelle nous souhaitons que S "obéisse" à sa table ($t_{n+1} = t_8 = 12 : 07$ dans notre exemple). Pour chacun des intervalles t_i à t_{i+1}, $i = 1, 2, \ldots, n$, définit un *état intervalle* (non maximal) qui est la "région" dans l'espace des phases composée de tous les états maximaux [...] avec $t_i \leq t \leq t_{i+1}$. (I.e., S est dans s_1 uniquement si S est dans un des états maximaux de cette "région"). Notons que le système S est dans s_1 de t_1 à t_2, dans s_2 de t_2 à t_3, ..., dans s_n de t_n à t_{n+1}. [...]
>
> Définissons $A = s_1 \vee s_3 \vee s_5 \vee s_7$; $B = s_2 \vee s_4 \vee s_6$.
>
> On vérifie aisément que S est dans l'état A de t_1 à t_2, de t_3 à t_4, et de t_5 à t_6, et de t_6 à t_7, et que S est dans l'état B à tous les temps entre t_1 et t_8. Ainsi S "a" la table de transition que nous avons spécifiée, et comporte les états A, B que nous venons de

> définir comme étant les 'réalisations' des états A, B décrits par la table de transition [Putnam, 1988, 122-123].

Pour résumer, Putnam considère un système physique arbitraire qui possède une dynamique continue, partitionne sa dynamique en des intervalles de temps discrets, et regroupe les parties du système afin qu'elles correspondent à une suite arbitraire d'états computationnels. Il conclut pour finir que tout sytème physique implémente n'importe quel automate fini.

Cependant, Putnam signale que son argument ne s'applique pas directement à *tous* les systèmes physiques. En particulier, des systèmes physiques tels que les systèmes cognitifs échappent à l'argument. Ces derniers reçoivent en effet des données en entrée physiques à travers leurs organes sensoriels et produisent des données en sortie physiques *via* leurs organes moteurs. Pour déterminer quels calculs sont implémentés par un système comportant à la fois des données en entrée et en sortie physiques, il est donc nécessaire de prendre en compte ces deux types de données. Voici ce que répond Putnam afin de renforcer son précédent argument :

> Imaginons [...] qu'un objet S qui prend en entrée des suites de symboles composées uniquement du symbole "1" et qui imprime des suites similaires en sortie, se comporte de 12 : 00 à 12 : 07 exactement comme s'il possédait une certaine description [computationnelle] D. Cela signifie d'une part que S reçoit une certaine suite, disons "111111", à 12 : 00 et imprime une certaine suite, disons "11", à 12 : 07, et d'autre part qu'il "existe" (d'un point de vue mathématique) une machine dont la description est D faisant de même (en étant dans l'état approprié à chacun des intervalles spécifiés, disons de 12 : 00 à 12 : 01, de 12 : 01 à 12 : 02, ..., [...]). Dans ce cas, S aussi peut être interprété comme étant dans les mêmes états logiques $A, B, C, \ldots$ aux mêmes temps et comme suivant les mêmes règles de transition ; c'est-à-dire, nous pouvons trouver des états *physiques* $A, B, C, \ldots$ qui seront les états de S aux temps appropriés et qui entretiendront des relations causales qui seront appropriées aux données en entrée et

> aux données en sortie. La démonstration est exactement la même [...] Nous obtenons ainsi que *la supposition selon laquelle quelque chose est une 'réalisation' d'une description d'un automate donné [...] est équivalente à l'énoncé selon lequel quelque chose se comporte comme s'il possédait une telle description* [Putnam, 1988, 124].

L'argument de Putnam consiste à considérer dans un premier temps un système physique arbitraire avec des données en entrée et en sortie physiques, et à le faire coïncider avec un automate fini arbitraire dont les données en entrée et en sortie abstraites sont mises en correspondance avec celles du système physique. Dans un second temps, il consiste à partitionner la dynamique interne du système physique comme dans l'argument précédent, et à reconstituer les parties afin qu'elles correspondent à la suite d'états computationnels de l'automate fini. Il suit qu'étant donnés un système physique et un automate fini arbitraires avec des données en entrée et en sortie, le système physique implémente l'automate.

Cet argument - et plus généralement les arguments en faveur du pancomputationnalisme fort - reposent implicitement ou explicitement sur le SMA (*simple mapping account*), à savoir sur l'analyse standard du calcul [Piccinini, 2010]. Ces arguments supposent en particulier qu'une application arbitraire d'une description computationnelle $\mathcal{C}$ vers la description physique d'un système est suffisante pour conclure que le système implémente $\mathcal{C}$. Le pancomputationnalisme faible est quant à lui défendu à l'aide d'arguments fondés sur des propositions alternatives à l'analyse standard :

- **L'analyse contrefactuelle** requiert que l'application entre les descriptions physiques et computationnelles soit telle qu'il existe un isomorphisme des relations contrefactuelles entre les états physiques vers les relations contrefactuelles entre les états computationnels [Maudlin, 1989].
- **L'analyse dispositionnelle** requiert que l'application soit telle qu'il existe un isomorphisme des relations dispositionnelles entre les états physiques vers les relations entre les états de transition qui sont spécifiés par la description computationnelle [Copeland, 1996], [Klein, 2008].

- **L'analyse causale** requiert que les relations entre les états physiques soient causales. Plus précisément, pour tout état de transition de la forme $s_1 \rightarrow s_2$ - spécifié par une description computationnelle - si le système est dans l'état physique appliqué à s_1 alors le système entre dans l'état physique appliqué à s_2 de façon causale [Chrisley, 1995], [Chalmers, 1995], [Scheutz, 2001].
- **L'analyse sémantique** requiert que l'application soit faite uniquement à partir des états physiques qui sont considérés comme des *représentations* [Fodor, 1975], [Cummins, 1983], [Pylyshyn, 1984], [Shagrir, 2006].
- **L'analyse syntaxique** requiert que l'application soit faite uniquement à partir des états physiques qui sont considérés comme *syntaxiques* [Stich, 1983].

Il n'est pas nécessaire de rentrer dans les détails de ces propositions, l'important étant de comprendre que ces propositions impliquent un pancomputationnalisme faible. Même si elles sont différentes de l'analyse standard, ces propositions conduisent en effet toutes à la thèse selon laquelle tous les systèmes physiques calculent. La raison en est que ces propositions possèdent la même structure définitionnelle que celle de l'analyse standard : elles ne sont fondées que sur l'ajout de restrictions supplémentaires soit sur l'application entre les états physiques et computationnels soit sur ces états proprement dits.

A présent que le pancomputationnalisme est défini, voici pourquoi il est utile pour résoudre le problème soulevé dans ce chapitre, à savoir le problème de la vérification.

1.2.2 Intérêt du pancomputationnalisme

L'analyse standard - qui repose sur la correspondance entre un système physique et une description computationnelle - conduit au pancomputationnalisme, thèse selon laquelle tous les systèmes physiques calculent. L'intérêt du pancomputationnalisme provient du fait qu'il peut servir de cadre théorique afin de résoudre le problème de la vérification, à savoir le problème de

déterminer si un système physique - et plus précisément un dispositif - hyper-calcule.

Premièrement, le pancomputationnalisme - qu'il soit fort ou faible - ne remet pas en cause l'hyper-calcul au sein des systèmes physiques. En effet, ces deux types de pancomputationnalisme définissent uniquement une *borne inférieure* à la puissance calculatoire des systèmes physiques. Cette borne inférieure implique plus précisément que tout système physique calcule au moins - par opposition à au plus - un ensemble de fonctions Turing-calculables. Tandis que le pancomputationnalisme faible affirme que les systèmes physiques calculent un sous-ensemble propre des fonctions Turing-calculables, le pancomputationnalisme fort affirme quant à lui qu'ils calculent l'ensemble des fonctions Turing-calculables. Le pancomputationnalisme laisse ainsi ouverte la possibilité pour un système physique de calculer au-delà de l'ensemble des fonctions Turing-calculables, c'est-à-dire d'hyper-calculer.

Deuxièmement, le pancomputationnalisme peut être considéré comme une première étape en vue d'attribuer l'hyper-calcul à un système physique car il permet d'affirmer qu'une HM physiquement construite - qui est un système physique - calcule. En effet, une HM est d'un point de vue structurel l'extension d'une MT. Par exemple, une *machine de Turing accélérante*, une *machine de Turing à temps infinis* ou une machine de Turing à oracle sont toutes des MT qui ont la capacité d'exécuter une opération supplémentaire - exécuter un *supertask* pour les deux premières et faire intervenir un oracle pour la troisième, *cf.* chapitre 1. Par conséquent, l'analyse standard permet d'établir qu'une HM physiquement construite calcule puisqu'une sous-partie de son dispositif implémente un automate fini, à savoir une MT.

Dans la suite de cette section, je souhaite déterminer si l'analyse standard fournit un cadre théorique pour affirmer qu'un système hyper-calcule. Plus précismément, mon idée est de me servir du critère proposé par l'analyse standard - à savoir l'existence d'une correspondance entre les états d'un système physique et ceux d'une description computationnelle - afin d'évaluer s'il est possible d'établir que certains systèmes physiques implémentent une HM.

1.3 L'analyse standard permet-elle de résoudre le problème de la vérification ?

Mon but est ici d'expliquer pourquoi l'analyse standard ne permet pas de résoudre le problème de la vérification, à savoir d'affirmer qu'un système physique hyper-calcule. Pour ce faire, ma stratégie n'est pas de montrer que l'analyse standard est incapable d'établir qu'un système hyper-calcule ; au contraire, je souhaite montrer que l'analyse standard peut conduire - sous certaines conditions - à un *hyper-pancomputationnalisme*, c'est-à-dire à la thèse selon laquelle tous les systèmes physiques hyper-calculent. En d'autres termes, je soutiens que l'analyse standard est inadéquate pour déterminer si un système physique hyper-calcule car elle peut avoir pour conséquence de considérer tous les systèmes physiques comme des HM.

L'analyse standard permet en effet selon moi d'attribuer aux systèmes physiques des hyper-calculs à partir de l'existence - au sens mathématique - d'une application entre les états de la description computationnelle et les états de la description physique du système. Plus précisément, l'analyse standard est fondée sur un présupposé implicitement admis : si un système physique $\mathcal{S}$ possède une description qui lui attribue une infinité non dénombrable d'états physiques alors il est toujours possible, à partir de classes d'équivalences, de construire une application des états physiques de $\mathcal{S}$ vers les états d'une machine abstraite $\mathcal{C}$, états qui sont par définition dénombrables. Ce présupposé est en particulier fondamental pour l'attribution du calcul aux systèmes physiques puisque ces derniers sont généralement décrits par des équations différentielles ayant pour solutions des nombres réels qui leur attribuent un nombre infini non dénombrable d'états.

Cependant, que se passerait-il si aucune classe d'équivalence ne permettait de faire correspondre l'infinité non dénombrable d'états physiques avec l'infinité dénombrable d'états computationnels ? Dans ce cas, il serait impossible pour la machine abstraite $\mathcal{C}$ d'être une MT car elle serait obligée de posséder un nombre non dénombrable d'états - contrairement à la MT qui en possède une quantité dénombrable. Plus encore, l'ajout d'un nombre infini non dénombrable d'états à une machine abstraite telle que la MT a pour

résultat de la faire évoluer en une machine capable d'hyper-calculer appelée
.

D'un point de vue structurel, une machine à états infinis () est une MT qui possède un nombre infini - dénombrable ou non - d'états internes et un nombre infini de transitions pouvant être exécutées par la machine lors d'un calcul [Ord, 2002]. D'un point de vue calculatoire, la stratégie permettant à la MEI de calculer des fonctions non Turing-calculables est identique à peu de choses près à celle de la machine de Turing accélérante - *cf.* chapitre 1. Plus précisément, l'exécution d'un nombre infini d'étapes de calcul qui permet à la machine de Turing accélérante d'hyper-calculer est remplacée par l'exécution d'un nombre infini de transitions. En d'autres termes, si la valeur d'une fonction nécessite l'exécution d'une infinité d'étapes pour être calculée, cette infinité d'étapes peut être effectuée par la MEI à partir du nombre infini de ses transitions.

A présent, supposons qu'un système comportant une infinité d'états physiques puisse posséder pour description computationnelle non pas une MT mais une MEI. Une telle supposition produirait immédiatement l'effet suivant : tout système physique dont la description est une équation différentielle implémenterait une MEI. En particulier, si un système physique $\mathcal{S}$ était décrit par une équation différentielle alors il existerait une application des états de la description physique de $\mathcal{S}$ vers les états internes d'une MEI tels que les états de transition de $\mathcal{S}$ correspondent aux états de transition de la MEI. Autrement dit, pour chaque état physique s du système $\mathcal{S}$ - autrement dit une infinité non dénombrable d'états - il existerait un état s_m de la MEI qui serait l'image de s par l'application. On pourrait alors conclure que tout système physique pouvant être décrit par une équation différentielle est capable d'hyper-calculer puisqu'il implémente une HM, à savoir une MEI.

On pourrait à première vue se satisfaire de ce résultat car il permettrait de disposer d'un critère, à savoir l'équation différentielle, pour déterminer si un système physique tel qu'une HM physiquement construite est capable d'hyper-calculer. En particulier, si ce résultat était exact alors la grande majorité des systèmes physiques seraient capables d'hyper-calculer puisque tout système décrit par une équation différentielle devrait être considéré comme

une HM. Le problème est que ce dernier résultat est en totale contradiction avec les recherches actuelles au sujet de la possibilité pour un système physique d'hyper-calculer. Ces recherches montrent en effet qu'il existe des positions très divergentes à ce sujet comme en témoignent les travaux de Kreisel [Kreisel, 1974], de Gandy [Gandy, 1980], de Pour-El & Richard [Pour-El and Richard, 1981], et de Weihrauch & Zhong [Weihrauch and Zhong, 2002]. Attribuer la capacité d'hyper-calculer à tous les systèmes physiques pouvant être décrits par des équations différentielles serait par conséquent incorrect.

En résumé, puisque les critères énoncés par l'analyse standard conduisent à trivialiser l'hyper-calcul au sein des systèmes physiques, cette analyse est d'après moi insuffisante pour déterminer si une machine physiquement construite est capable de calculer des fonctions non Turing-calculables.

Plus généralement, cette insuffisance a pour origine l'incapacité de l'analyse standard à déterminer *ce* que calcule un système physique. Autrement dit, si l'analyse standard est capable d'affirmer qu'un système calcule, déterminer si ce calcul est effectif ou non lui est impossible. Cela s'explique par le fait que l'analyse standard a pour but de déterminer quels sont les systèmes physiques qui calculent indépendamment du type - Turing-calculable ou non - des fonctions mathématiques qui sont calculées. Qu'un système physique calcule de façon effective ou hyper-calcule n'est pas de grande importance puisque d'après l'analyse standard le système exécute au moins un calcul dans les deux cas.

Puisque l'analyse standard ne permet pas de résoudre le problème de la vérification, il est par conséquent nécessaire d'utiliser des critères différents de ceux inscrits dans l'analyse standard pour affirmer qu'un système physique est une HM. Plus précisément, je propose d'analyser dans la section suivante la seconde approche énoncée en introduction qui consiste à étudier non pas les systèmes physiques en général mais uniquement les ordinateurs. Cette approche est en particulier fondée sur la conception de techniques de vérification - théoriques et empiriques - appliquées aux parties logicielle - programme - et matérielle - composants électroniques par exemple - des ordinateurs. Mon but sera ensuite (1) de me servir de ces techniques de vérification pour établir des critères permettant de vérifier - dans un sens précis - les valeurs fournies

par les ordinateurs ; et (2) d'évaluer si ces critères peuvent être appliqués à l'hyper-calcul afin d'affirmer que les valeurs fournies par un dispositif sont celles d'une fonction non Turing-calculable.

2 Les ordinateurs face au problème de la vérification

Dans cette section, je vais montrer qu'une approche fondée sur l'utilisation de techniques - à la fois théoriques et empiriques - permet de résoudre *partiellement* le problème de la vérification appliqué aux ordinateurs. Partiellement car ces techniques de vérification ne permettent pas de prouver qu'un ordinateur calcule une fonction donnée ; elles permettent uniquement d'augmenter la probabilité selon laquelle l'ordinateur calcule cette fonction.

Si le problème de la vérification n'est pas résoluble *stricto sensu*, cette résolution est néanmoins suffisante pour affirmer qu'un ordinateur calcule. On développe en effet une solide confiance dans les calculs exécutés par les ordinateurs, et cela même si de tels calculs ne peuvent pas être parfaitement vérifiés. Autrement, pourquoi les transports aériens ou la production d'énergie - en particulier celle d'origine nucléaire - reposeraient-ils sur l'utilisation d'ordinateurs ?

Enfin, puisque qu'une vérification partielle des résultats fournis par les ordinateurs est suffisante pour affirmer qu'ils calculent, pourquoi n'en serait-il pas de même pour les HM ? A l'aide des techniques de vérification, il pourrait être en particulier possible d'affirmer avec une forte probabilité qu'une machine physiquement construite calcule une fonction non Turing-calculable.

Avant d'évaluer si l'on peut appliquer ces techniques aux HM - but des sections suivantes - j'explique dans ce qui suit (1) pourquoi le problème de la vérification appliqué aux ordinateurs n'est pas résoluble *stricto sensu* ; et (2) pourquoi ce problème peut néanmoins être résolu partiellement. Pour ce faire, je commence par expliquer ce que signifie résoudre partiellement le problème de la vérification appliqué aux ordinateurs. Je montre ensuite que certaines techniques fondées sur le comportement entrée-sortie des ordinateurs sont

inefficaces pour résoudre ce problème ; tandis que d'autres - fondées sur les parties logicielle et matérielle des ordinateurs - permettent d'affirmer qu'un ordinateur calcule une fonction donnée.

2.1 Résoudre *partiellement* le problème de la vérification appliqué aux ordinateurs

Le problème de la vérification appliqué aux ordinateurs est énoncé de la façon suivante : soient f une fonction définie sur $\mathbb{N}$ qui est Turing-calculable et $\mathcal{O}$ un ordinateur, peut-on vérifier que $\mathcal{O}$ calcule f ? Mais que signifie *vérifier* que O calcule f ? La définition ci-dessous ne prend en compte que les fonctions unaires Turing-calculables mais cette dernière peut être généralisée aux fonctions n-aires Turing-calculables.

Définition (Vérification des résultats)
Soient f une fonction unaire définie sur $\mathbb{N}$ qui est Turing-calculable et $\mathcal{O}$ un ordinateur. Vérifier que $\mathcal{O}$ calcule f signifie être capable de vérifier que $f(x) = \mathcal{O}(x)$ pour tout $x \in \mathbb{N}$. Soit $x_0 \in \mathbb{N}$, on est capable de vérifier que $f(x_0) = \mathcal{O}(x_0)$ si et seulement si on est capable d'affirmer que $\mathcal{O}(x_0) = f(x_0)$.

L'affirmation selon laquelle $\mathcal{O}(x_0) = f(x_0)$ peut s'effectuer de deux manières différentes. Elle peut d'une part s'effectuer *via* une procédure effective qui calcule f ; on suit chacune des étapes de la procédure, de la donnée initiale au résultat, et on compare le résultat avec celui fourni par l'ordinateur. Elle peut d'autre part être réalisée *via* le programme de l'ordinateur si on possède au préalable le résultat du calcul ; on suit dans ce cas chacune des étapes du programme, de la donnée initiale au résultat, et on compare le résultat avec celui que l'on possède.

Si les conditions de vérification des résultats fournis par les ordinateurs viennent d'être énoncées, ces dernières ne sont toutefois satisfaites qu'en principe, à savoir en faisant abstraction des ressources physiques utilisées lors des calculs. En effet, ces conditions stipulent que le calculateur humain doit suivre chacune des étapes du calcul de f - *via* une procédure effective ou le programme de l'ordinateur - afin d'affirmer que les résultats fournis par

l'ordinateur sont égaux aux valeurs de f. Cette condition est ainsi satisfaite en principe puisque la possibilité de suivre les étapes du calcul de f - qui est par définition Turing-calculable - fait partie des contraintes imposées à la notion de procédure effective : « un être humain doit pouvoir suivre la procédure étape par étape, de la donnée initiale au résultat indépendamment des contraintes de temps et d'espace mémoire » [Copeland, 2002a, p. 1].

Ces conditions ne sont toutefois pas satisfaites en pratique, lorsque les ressources utilisées lors des calculs sont prises en compte. Prenons l'exemple du calcul de la fonction d définie par $d(n) =$ la n-ième décimale de l'expansion de π. Supposons par ailleurs que l'on dispose d'un ordinateur dont le programme approxime les décimales de π à l'aide de la *formule de Leibniz*

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Peut-on vérifier en pratique que cet ordinateur calcule d ? Autrement dit, est-il possible de vérifier que l'ordinateur fournit une valeur correcte pour un des arguments de la fonction ? Oui si les valeurs à vérifier sont comprises entre 1 et 20 car il est alors possible, avec de la patience, de suivre manuellement les étapes exécutées par l'ordinateur conduisant aux résultats. Cependant, dans le cas où l'on souhaite vérifier que la décimale de π récemment calculée par ordinateur pour $n = 10^{12}$ est bien égale à 5, la vérification manuelle est impossible à cause du manque de temps. Par conséquent, il est impossible en pratique de vérifier les résultats calculés par un ordinateur pour certains arguments d'une fonction, résultats qui demandent trop de ressources pour être vérifiés.

Même si la démonstration précédente implique que le problème de la vérification est insoluble - au sens où les résultats fournis par les ordinateurs sont en pratique impossibles à vérifier - on ne remet généralement pas en cause le fait que les ordinateurs calculent. Dans le cas contraire, pourquoi les ordinateurs seraient-ils la base de nos réseaux de transports, de nos systèmes d'énergie et de notre économie ? Il semble donc que l'on développe une solide confiance dans les calculs exécutés par les ordinateurs, et cela même si de tels calculs sont en pratique impossibles à vérifier.

Mais quelle est l'origine de cette confiance liée aux calculs exécutés par les ordinateurs ? Cette confiance est fondée sur la thèse selon laquelle *le calcul ne présuppose pas la vérification* au sens où il est possible d'affirmer de façon plausible qu'une fonction donnée est calculée par un ordinateur quand bien même une parfaite vérification est impossible [Shagrir and Pitowsky, 2003, p. 90]. Plus précisément, cette confiance résulte de l'utilisation de certaines méthodes à la fois empiriques et théoriques - méthodes qui seront par la suite explicitées - permettant d'augmenter la probabilité qu'un ordinateur calcule une fonction donnée.

En pratique, les résultats fournis par les ordinateurs ne peuvent donc pas être *parfaitement* vérifiés mais uniquement *partiellement* vérifiés, au sens où la vérification dépend du nombre et de la qualité des méthodes que l'on utilise. Autrement dit, vérifier partiellement qu'un ordinateur calcule une fonction f ne nécessite plus de pouvoir suivre chaque étape de ses calculs mais de pouvoir affirmer avec une forte probabilité que l'ordinateur calcule f.

Définition (Vérification partielle des résultats)
Soient f une fonction unaire définie sur $\mathbb{N}$ qui est Turing-calculable et $\mathcal{O}$ un ordinateur. Vérifier partiellement que O calcule f signifie être capable de vérifier partiellement que $f(x) = \mathcal{O}(x)$ pour tout $x \in \mathbb{N}$. Soit $x_0 \in \mathbb{N}$, on est capable de vérifier partiellement que $f(x_0) = \mathcal{O}(x_0)$ si et seulement si on affirme avec une forte probabilité que $\mathcal{O}(x_0) = y$.

Pour affirmer avec une forte probabilité qu'un ordinateur calcule une fonction particulière, les scientifiques ne peuvent pas se reposer sur une étude du comportement entrée-sortie des ordinateurs ; ils sont au contraire obligés d'étudier leur fonctionnement interne qui est divisé en deux parties : une partie logicielle - programmes - et une partie matérielle - composants électroniques.

2.2 Vérification par identification

Mon but est ici d'expliquer pourquoi les scientifiques sont obligés d'avoir accès au fonctionnement interne des ordinateurs pour vérifier leurs résultats.

Plus précisément, je souhaite montrer qu'il est impossible d'affirmer qu'un ordinateur calcule une fonction donnée si l'on se base uniquement sur son fonctionnement externe, à savoir son comportement entrée-sortie - les données en entrée qu'on lui fournit et les données en sortie qu'il produit.

Si l'on ne dispose que du fonctionnement externe des ordinateurs, le problème de la vérification est transformé en un problème plus général : *le problème de l'identification*. En effet, pour vérifier les valeurs fournies par un ordinateur il est nécessaire de savoir au préalable à quelle fonction correspondent ces valeurs ; autrement, comment vérifier que l'ordinateur calcule une fonction particulière ? Ici, puisque l'on ne dispose que du comportement entrée-sortie de l'ordinateur, le problème ne consiste donc pas à vérifier si les valeurs produites par ce dernier correspondent à une fonction particulière, mais il consiste à identifier la fonction qui correspond à ces valeurs.

Pour montrer qu'il est impossible d'affirmer qu'un ordinateur calcule une fonction donnée si l'on se base uniquement sur son comportement entrée-sortie, je vais par conséquent expliquer pourquoi le problème de l'identification n'est pas résoluble.

2.2.1 L'inférence effective

Tenter d'identifier une fonction à partir d'un ensemble de données en entrée et en sortie peut être considéré comme une forme d'*inférence effective* dont le but est de produire une règle effective et générale sur la base d'un nombre fini de données entrée-sortie [Angluin and Smith, 1983]. En effet, les valeurs produites par une telle règle à partir de l'ensemble des données entrée-sortie d'un ordinateur correspondent aux valeurs d'une fonction Turing-calculable particulière ; l'inférence effective permet ainsi d'identifier la fonction qui est calculée par un ordinateur.

Malheureusement, le problème de l'inférence effective est insoluble. Avant de rentrer dans les détails formels de cette affirmation, expliquons intuitivement pourquoi ce problème est insoluble. Considérons par exemple le comportement entrée-sortie suivant où $f(x)$ est le résultat de l'ordinateur avec

l'entrée x :

$$f(1) = 2, f(2) = 4, f(3) = 6 \ \ldots$$

A partir de ces résultats, est-il possible d'inférer que l'ordinateur calcule la multiplication par deux ? Non car même si l'on observe que chaque nombre testé est multiplié par 2, il sera toujours possible que 3^{115} soit multiplié par 3 et donc que l'inférence selon laquelle la boîte multiplie par 2 est fausse. Autrement dit, identifier la fonction qui est calculée par la boîte noire est impossible car il existera toujours plusieurs opérations qui seront cohérentes avec la quantité finie d'observations que l'on aura collectée [Kripke, 1982].

La démonstration formelle de ce résultat est cependant plus complexe. Tout d'abord, définissons précisément le problème de l'inférence effective [Adleman and Blum, 1991, p. 892]. Soit M une machine - par exemple une MT - composée de trois rubans infinis : un ruban de lecture pour lire les données en entrée, un ruban d'écriture pour écrire les données en sortie et un ruban de travail pour exécuter les étapes intermédiaires. Une fonction f récursive totale est ensuite représentée sur son premier ruban par des couples $(x_n, f(x_n))$ pour $n \in \mathbb{N}$. Ces couples sont rangés par ordre croissant de sorte que le premier couple soit $(x_0, f(x_0))$ puis le second $(x_1, f(x_1))$ et ainsi de suite - la fonction est définie pout tout $n \in \mathbb{N}$ puisque f est totale. Enfin, M *infère* f si et seulement si les quatre conditions suivantes sont satisfaites :

1. Chaque nombre inscrit sur son ruban de lecture est éventuellement lu par M.
2. M produit en sortie un nombre infini de nombres - un à la fois.
3. Presque tous les nombres produits en sortie sont identiques, disons au nombre n.
4. n est l'index de f, c'est-à-dire $\phi_n = f$.

Intuitivement, M tente d'identifier la règle qui produit la fonction représentée sur son ruban de lecture. Toutefois, puisque M ne peut lire qu'une partie finie de l'ensemble des valeurs de la fonction, cette identification ne peut s'effectuer qu'à partir d'une quantité partielle d'information. On autorise par conséquent la machine à proposer plusieurs réponses afin qu'elle puisse améliorer sa réponse en fonction de la quantité d'information qu'elle reçoit.

On dit alors que *M infère f* si et seulement si elle propose la bonne réponse et garde cette réponse indéfiniment.

Le problème de l'inférence effective peut s'énoncer de la façon suivante : existe-t-il une machine M - c'est-à-dire une procédure effective - capable d'inférer toutes les fonctions récursives totales, à savoir toutes les fonctions calculables par les ordinateurs ? Si une telle machine M existait, le problème de l'identification serait résolu puisque M serait capable d'inférer - et donc d'identifier - les fonctions calculées par un ordinateur. Gold a néanmoins répondu à cette question par la négative en démontrant le théorème suivant [Gold, 1967] :

Théorème (Gold)
Pour toute machine M, il existe une fonction récursive totale f telle que M n'infère pas f.

Une des méthodes pour démontrer ce résultat consiste à prouver que le problème de l'inférence effective est équivalent au problème de l'arrêt propre aux MT [Adleman and Blum, 1991]. Le résultat de Gold signifie en outre qu'il est impossible d'inférer, à partir d'un comportement entrée-sortie d'une machine, la fonction qui est calculée par cette machine où f est une fonction récursive totale.

Plus généralement, si l'on ne dispose que du comportement entrée-sortie d'une machine, les problèmes consistant à inférer certaines caractéristiques que possède une fonction calculée par cette machine sont souvent indécidables. Considérons en particulier un cas assez similaire au problème précédent mais qui est lié à l'*inférence prédictive*.

2.2.2 L'inférence prédictive

L'inférence prédictive est utilisée pour modéliser le processus par lequel un scientifique est capable de prédire le résultat d'une future observation - et non pas une règle générale - sur la base d'un nombre fini d'observations. Ce problème est défini de la même manière que le problème de l'inférence effective sauf en ce qui concerne les deux dernières conditions à satisfaire. Ici, *M prédit f* si et seulement si les conditions suivantes sont satisfaites :

1. Chaque nombre inscrit sur son ruban de lecture est éventuellement lu par M.
2. M produit en sortie un nombre infini de nombres (un à la fois).
3. M produit sa donnée en sortie numéro i, notée $M_f(i)$, avant de lire la donnée en entrée numéro i notée $f(i)$.
4. Les $M_f(i)$ sont des variantes finies des $f(i)$.

Intuitivement, M lit à l'étape n les nombres $f(0), f(1), \ldots, f(n-1)$ et essaie de prédire $f(n)$. Après avoir proposé sa réponse, la machine peut aller à l'étape $n+1$ où elle pourra lire la bonne réponse $f(n)$ et déterminer si sa dernière réponse était correcte ou non. Comme dans le problème précédent, les réponses incorrectes sont tolérées ; M prédit f si et seulement s'il existe une étape n pour laquelle M commence à produire les bonnes réponses et à partir de laquelle plus aucune mauvaise réponse n'est produite.

Ici encore, le problème de déterminer s'il existe une machine pouvant prédire toute fonction f récursive totale ne possède pas de réponse positive [Gold, 1967] :

Théorème (Gold)
Pour toute machine M, il existe une fonction récursive totale f telle que M ne prédit pas f.

Ces deux résultats négatifs à propos des inférences effective et prédictive ont été présentés dans le but de montrer que le problème de la vérification ne pouvait pas être résolu uniquement à partir du fonctionnement externe d'un ordinateur - son comportement entrée-sortie. Pour résoudre - du moins partiellement - le problème de la vérification, il est au contraire nécessaire d'étudier la structure interne des ordinateurs. Voici une présentation des techniques de vérification fondées sur leurs parties logicielle et matérielle. Ces techniques permettent en particulier de vérifier partiellement les résultats fournis par les ordinateurs, autrement dit d'affirmer avec une forte probabilité qu'un ordinateur calcule une fonction donnée.

2.3 Vérifications logicielle et matérielle

Il existe de nombreuses méthodes permettant de développer une confiance suffisante afin d'affirmer qu'un ordinateur calcule une fonction particulière. Ces méthodes sont divisées en deux catégories : celles portant sur la vérification du programme implémenté dans l'ordinateur - la vérification logicielle - et celles portant sur une vérification des composants matériels de l'ordinateur - la vérification matérielle.

La vérification logicielle a pour but de vérifier que le programme permettant à un ordinateur de calculer une fonction donnée est correct [Kurshan, 2000]. Une telle vérification peut s'effectuer de deux manières différentes : de manière empirique en exécutant le programme pour des données en entrée arbitraires ou de manière formelle en utilisant des outils mathématiques. La première méthode consiste à déceler les problèmes inhérents aux programmes - appélés *bugs* - en les testant pour des millions de données en entrée dites *typiques* - qui sont susceptibles d'être fournies au programme lors de son utilisation. Toutefois, les limites de cette méthode sont claires : puisque le nombre de données en entrée à tester est infini, « tester un programme peut servir à montrer la présence de bugs, mais ne pourra jamais servir à montrer leur absence ! » [Dahl et al., 1972]. C'est pourquoi la vérification d'un programme est complétée par une seconde méthode - appelée *vérification formelle* - qui n'est pas fondée sur l'exécution du programme.

La vérification formelle tente de prouver mathématiquement qu'un ordinateur calcule une fonction donnée [Barwise, 1989]. Elle consiste à produire une preuve mathématique de la conformité entre une machine devant calculer une fonction donnée - l'ordinateur - et une spécification du calcul qu'elle doit exécuter - le programme. Malheureusement, l'écriture d'une telle preuve comporte la plupart du temps de nombreuses erreurs du fait de sa longueur. Ces erreurs rendent ainsi nécessaire une vérification systématique de la preuve qui oblige généralement le mathématicien à passer plus de temps sur la vérification de la preuve que sur sa conception. Ce dernier problème a été néanmoins minimisé grâce à l'utilisation d'ordinateurs dont le programme a pour objectif de vérifier - et même parfois de produire - la preuve en ques-

tion [Boyer and Moore, 1984]. En particulier, la vérification par ordinateurs possède deux vertus : elle permet d'une part de réduire les risques d'erreurs en vérifiant que les moyens utilisés par le mathématicien conduisent à des preuves correctes et permet d'autre part de rendre plus claires certaines parties de la preuve en diminuant si possible la longueur de ses parties.

La vérification formelle n'est cependant pas suffisante pour affirmer qu'un ordinateur calcule une fonction donnée. En effet, puisque cette vérification est fondée sur la thèse selon laquelle un programme exécuté par un ordinateur peut être vérifié par un autre ordinateur, cela conduit immédiatement à la question de la vérification du programme exécuté par le second ordinateur. Il est par conséquent nécessaire de fonder la vérification sur une partie différente de la partie logicielle si l'on souhaite affirmer qu'un ordinateur calcule une fonction donnée.

La vérification matérielle comporte deux approches : une approche logique et une approche physique. L'approche logique peut être illustrée à partir du problème de *la comparaison combinatoire*, à savoir le problème de vérifier si deux circuits électroniques possèdent le même comportement calculatoire. Résoudre ce problème permettrait en particulier de vérifier que deux ordinateurs ayant des architectures internes différentes calculent les mêmes fonctions. L'approche logique consiste plus précisément à formaliser un problème tel que la comparaison combinatoire et de le résoudre à partir d'outils logiques. Par exemple, déterminer si deux circuits ont le même comportement calculatoire peut être réduit au problème de vérifier si une formule de la logique propositionnelle est une tautologie [2].

Pour vérifier qu'une proposition est une tautologie, il suffit simplement d'exécuter la procédure effective suivante : (1) construire la table de vérité correspondant à cette formule ; et (2) vérifier si cette dernière prend toujours la valeur vrai. Cependant, bien que cette procédure soit toujours possible en principe - la table de vérité est par définition finie - elle n'est pas toujours

2. En logique propositionnelle, une tautologie est une proposition qui est vraie quelle que soit la valeur de vérité de ses composants élémentaires. Dit autrement, la table de vérité correspondant à cette proposition prend toujours la valeur vrai. D'après Kleene, c'est à la suite du *Tractacus Logico Philosophicus* [Wittgenstein, 1921] que l'on a appelé *tautologie* ce type de proposition [Kleene, 1967].

réalisable en pratique lorsque la proposition comprend une grande quantité de composants élémentaires. En effet, tous les algorithmes connus pour résoudre ce problème ont un temps d'exécution exponentiel en fonction de la taille de l'entrée dans le pire cas, et sont donc inexploitables en pratique même pour des instances de taille modérée [Davis and Putnam, 1960].

Enfin, l'approche physique peut s'effectuer soit sur le matériel proprement dit soit sur les théories physiques qui régissent le matériel. D'une part, tester le matériel se fait généralement à partir de calculs exécutés en parallèle par un ensemble d'ordinateurs identiques tant au niveau de la programmation qu'au niveau du matériel. De cette manière, si un ordinateur ne produit pas la même réponse que le groupe cela voudra dire avec une forte probabilité qu'un de ses composants est défectueux. D'autre part, il est possible d'augmenter la probabilité qu'un ordinateur se comporte correctement à partir des connaissances que l'on possède des théories physiques qui régissent le fonctionnement du matériel de l'ordinateur - par exemple ses circuits électroniques. En d'autres termes, le comportement calculatoire d'un ordinateur peut être expliqué à partir des théories physiques sur lesquelles est fondé son matériel :

> Nous pouvons utiliser nos théories physiques pour expliquer quelle fonction est calculée. Nos théories physiques permettent d'établir comment un dispositif se comportera d'après un ensemble d'équations [Shagrir and Pitowsky, 2003, p. 91].

En résumé, l'utilisation conjointe des techniques qui viennent d'être présentées - techniques fondées à la fois sur les programmes et le matériel informatiques - permettent d'augmenter la confiance que l'on confère aux ordinateurs. Cette confiance, si elle est insuffisante pour démontrer qu'un ordinateur calcule une fonction particulière, est néanmoins suffisante pour l'affirmer. Par conséquent, le problème de la vérification appliqué aux ordinateurs est résoluble *à condition* de considérer une vérification *partielle*. Dans le cas où cette condition est acceptée, il devient possible de vérifier qu'un ordinateur calcule une fonction Turing-calculable f donnée à partir du moment où la confiance relative au calcul de f est suffisante pour affirmer que l'ordinateur calcule f.

Puisque le problème de la vérification appliqué aux ordinateurs peut

être résolu *via* la notion de vérification partielle, pourquoi n'en serait-il pas de même pour le problème de la vérification appliqué aux HM ? Plus précisément, il se pourrait que l'utilisation de techniques similaires à celles utilisées dans le cas des ordinateurs permette d'affirmer qu'une HM physiquement construite calcule une fonction non Turing-calculable. La vérification des résultats fournis par les HM serait donc fondée sur la confiance que l'on confère aux hyper-calculs exécutés par ces machines.

Avant d'évaluer si les techniques présentées ci-dessus sont suffisantes pour vérifier partiellement les résultats fournis par les HM, je souhaite expliquer d'où provient la possibilité de vérifier les résultats fournis par un ordinateur. Cette explication permettra de formuler un ensemble de critères qui devra être satisfait pour pouvoir affirmer que les résultats fournis par une machine sont partiellement vérifiables, que cette machine soit un ordinateur ou une HM.

2.4 L'utilisabilité du calcul

Si l'on considère une vérification partielle, la possibilité de vérifier les résultats fournis par les ordinateurs a selon moi pour origine une de leurs caractéristiques propres : leur *utilisabilité*. Je pense en effet que c'est parce que les ordinateurs sont utilisables que l'on peut se servir des techniques théoriques et expérimentales qui ont été présentées ci-dessus afin d'augmenter la confiance que l'on confère à leurs calculs. Ce passage qui va suivre aura donc un double but : définir d'une part ce qu'est l'utilisabilité d'un ordinateur et montrer d'autre part pourquoi cette utilisabilité est nécessaire à la vérification partielle des calculs exécutés par les ordinateurs.

2.4.1 Les conditions d'utilisabilité

Pour définir l'utilisabilité d'une machine, je vais m'appuyer en grande partie sur une définition de l'utilisabilité proposée par Piccinini [Piccinini, 2011, p. 740]. Pour ce dernier néanmoins, l'utilisabilité ne concerne pas les machines proprement dites mais les calculs. En effet, l'utilisabilité est pour Piccinini une condition que doit satisfaire tout processus physique pour être

considéré comme un calcul. Cette condition est définie à partir de la notion d'observateur fini qui doit être prise dans son sens le plus large en englobant à la fois les êtres humains et les autres êtres intelligents - hypothétiques ou réels - qui possèdent des capacités finies :

Définition (Condition d'utilisabilité)
Si un processus physique est un calcul, il peut être utilisé par un observateur fini pour obtenir les valeurs d'une fonction.

Plus précisément, un processus physique est utilisable d'après Piccinini si

> Un observateur fini peut exécuter le processus afin de produire les valeurs d'une fonction sous la forme de données en sortie lisibles par l'observateur. Cela requiert de la part de l'observateur de pouvoir identifier quelle fonction est calculée, de pouvoir lire les données en entrée et en sortie, et d'être capable de construire le système qui exhibe le processus. Un processus physique utilisable est aussi un processus qui, de la même manière qu'une procédure effective, peut être en principe reproduit si un observateur fini souhaite redémarrer le processus [Piccinini, 2011, p. 741].

La définition que propose Piccinini peut être détaillée point par point à partir d'un ensemble de contraintes inscrites au sein de la condition d'utilisabilité. Un processus physique ou un calcul est dit utilisable s'il satisfait les contraintes suivantes :

1. **Les données en entrée et en sortie doivent être lisibles**. Une donnée en entrée ou en sortie est lisible si elle peuvent être lues par un observateur à un degré d'approximation arbitraire. En particulier, les données en entrée et en sortie des ordinateurs actuels sont lisibles car ces données sont des suites de symboles, à savoir des instantiations concrètes des lettres d'un alphabet fini [Piccinini, 2007].
2. **La fonction qui est calculée doit être définissable indépendamment du processus qui la calcule.** D'après Piccinini, le calcul d'une fonction suppose que l'on puisse déterminer quelle est la fonction qui est calculée par le processus avant de le faire fonctionner. Une machine

dont seul le comportement entrée-sortie est accessible ne peut donc pas être considérée comme utilisable puisqu'il est impossible de déterminer la fonction qui est calculée [Gold, 1967].

3. **Le processus doit en principe pouvoir être reproduit**. Il doit pouvoir être reproduit par tout observateur souhaitant obtenir les résultats fournis par le processus. Cette dernière phrase doit être expliquée plus en détail car une suite d'évènements physiques ne peut généralement pas se produire deux fois à l'identique. Les processus physiques peuvent toutefois être reproduits lorsque les mêmes suites d'états - comme ceux définis à partir d'un ensemble d'équations dynamiques - sont reproduits soit au sein du même système physique à différents moments, soit au sein de deux systèmes physiques similaires qui satisfont par exemple les mêmes équations. Un système physique est donc reproductible si l'observateur est capable de configurer un système physique qui suit la même suite d'états de transition que le processus d'origine.
4. **Le processus doit pouvoir être configuré.** Un système physique capable de calculer - un ordinateur par exemple - est utilisable si l'utilisateur de ce système peut le configurer en lui fournissant pour données en entrée les différents arguments de la fonction qui est calculée par le système. Lorsque l'utilisateur souhaite commencer un nouveau calcul, le système doit aussi pouvoir être réinitialisé pour qu'un nouvel argument - qui peut être identique au précédent - puisse lui être fourni. Si la donnée en entrée reste inchangée alors le résulat du calcul doit lui aussi demeurer inchangé. En résumé, un système ou une machine est configurable si l'utilisateur peut choisir la donnée en entrée sur laquelle le calcul se portera.

D'après Piccinini, la condition d'utilisabilité qui vient d'être présentée a pour but de caractériser les processus physiques pouvant être considérés comme des calculs. Même si je vais m'appuyer sur cette condition, la condition d'utilisabilité que je compte définir n'aura pas pour fonction de caractériser les processus physiques pouvant être considérés comme des calculs. Plus précisément, elle aura pour fonction de caractériser les machines

qui sont partiellement vérifiables. Autrement dit, si une machine est utilisable cela signifiera que les résultats fournis par cette machine peuvent être partiellement vérifiés.

Dans le but de défendre cette dernière affirmation, je vais compléter la définition proposée par Piccinini en ajoutant deux contraintes supplémentaires.

Définition ()

Une machine physiquement construite est utilisable si et seulement si elle satisfait les contraintes suivantes :

1. *. Les données en entrée et en sortie doivent être lisibles.*
2. *. La machine doit pouvoir être configurée.*
3. *. Les calculs doivent pouvoir être reproduits.*
4. *. Les fonctions calculées doivent pouvoir être définies indépendamment de la machine qui les calcule.*
5. *. Les résultats fournis par la machine doivent être accessibles.*
6. *. Les composants de la machine doivent être fiables.*

Dans cette nouvelle définition, les contraintes de lisibilité, de configuration, de reproduction et de définissabilité sont identiques à celles qui ont été proposées par Piccinini. Les contraintes d'accès et de fiabilité ont quant à elles été ajoutées. Voici une explication détaillée de ces deux contraintes.

Accès. La contrainte d'accès nécessite de la part de l'utilisateur de pouvoir avoir accès aux résultats fournis par la machine. Pour comprendre en quoi l'accès aux résultats est une condition nécessaire à l'utilisabilité d'une machine, considérons le cas des ordinateurs. les ordinateurs sont construits pour accéder aux valeurs de fonctions qui ne peuvent pas être calculées manuellement - *cf.* chapitre 2. Une machine dont les résultats seraient inaccessibles perdrait son utilité première, à savoir l'exécution de calculs infaisables en pratique par un calculateur humain. Elle devrait donc être considérée comme étant inutilisable.

Il est toutefois important de comprendre que la contrainte d'accès est différente de la contrainte de lisibilité des données en entrée et en sortie. Cette affirmation peut être illustrée à partir du passage suivant :

> En tant qu'expérience de pensée, nous pourrions imaginer une machine de Turing qui calcule une fonction calculable et qui traverse en même temps l'horizon des évènements d'un trou noir. Dans ce cas, il n'y a pas de doute qu'une fonction sera évaluée, mais il n'existe en principe aucun moyen qui permettent de découvrir les résultats du calcul [Duwell, 2004, p. 75].

Duwell explique que l'on peut concevoir - du moins mentalement - une MT qui fournit en sortie des résultats lisibles mais inaccessibles. Ces résultats sont lisibles car ce sont par définition des suites de symboles pouvant être lus à un degré d'approximation choisi ; ils ne sont en revanche pas accessibles au sens où l'utilisateur ne peut pas les observer. Dans le cas de la MT décrite par Duwell, l'utilisateur ne peut donc pas vérifier - même de manière partielle - que le calcul a bien été effectué puisque les valeurs de la fonction qui est calculée lui sont inacessibles. La contrainte d'accès est par conséquent un prérequis indispensable si l'on souhaite vérifier les résultats fournis par une machine.

Fiabilité. La dernière contrainte faisant partie de la condition d'utilisabilité est la fiabilité des composants de la machine. S'il est vrai que demander à une machine de ne jamais avoir de problèmes physiques serait irréaliste, une machine qui ne pourrait jamais fournir les résultats de ses calculs ou qui les altèrerait à cause de problèmes techniques liés à ses composants serait inutilisable : « une machine à calculer dont la fiabilité ne peut pas être vérifiée est inutile ! » [Copeland, 2004, p. 256]. Pour être utilisable, une machine doit donc pouvoir travailler correctement pendant un temps assez long dans le but de fournir le résultat de son calcul. Plus précisément, deux critères doivent être remplis pour que les composants d'une machine soient fiables : ils doivent être robustes - ne pas se détériorer facilement - et doivent être tels que les perturbations externes ne puissent pas conduire à une modification des résultats.

Le lien entre fiabilité des composants d'une machine et son utilisabilité peut être illustré à partir d'un exemple historique [Piccinini, 2011, p. 746]. Lorsque les ordinateurs ont été initialement proposés, une critique majeure

les concernant était portée sur la faible résistance des tubes à vide qui les composaient. En particulier, cette faiblesse liée aux tubes à vide des premiers ordinateurs les rendait presque inutilisables car les calculs qu'ils exécutaient demandaient l'aide à plein temps de techniciens dont le travail était de réparer ces composants. Cette critique adressée à l'égard des ordinateurs n'est toutefois plus pertinente puisque les ordinateurs actuels sont remarquablement fiables, ce qui les a rendu plus utilisables et donc plus accessibles à un large public.

2.4.2 Utilisabilité et vérification partielle

Si j'ai défini explicitement les contraintes qui constituent la condition d'utilisabilité d'une machine, il me reste néanmoins à expliquer pourquoi la vérification partielle des résultats fournis par une machine nécessite de satisfaire ces contraintes. Plus précisément, mon but est de montrer que si ces contraintes ne sont pas satisfaites, alors les méthodes de vérification partielle que j'ai présentées précédemment ne peuvent pas être utilisées. Voici un récapitulatif de ces méthodes :

1. Identification des bugs en exécutant le programme d'un ordinateur de manière répétée.
2. Création d'une preuve formelle établissant la correction du programme.
3. Identification des problèmes physiques liés au matériel en utilisant des ordinateurs travaillant en parallèle.
4. Explication du comportement d'un ordinateur *via* la théorie physique sur laquelle est fondée son fonctionnement.

Commençons tout d'abord par la contrainte de reproduction. Si les calculs ne peuvent pas être reproduits, ni l'identification de bugs ni les tests effectués en parallèle ne peuvent pas être réalisés. Le principe central sur lequel sont fondées ces deux méthodes est la répétition de l'exécution du programme. Pour l'identification de bugs, la répétition est nécessaire pour repérer les erreurs inscrites dans l'écriture du programme. Autrement dit, plus un calcul est exécuté et plus la probabilité de trouver des erreurs au

sein de sa spécification - son programme - augmente. Pour le calcul en parallèle, c'est la répétition des calculs qui permet d'établir qu'un ordinateur a un problème technique lié à son matériel. En effet, si on ne pouvait réaliser qu'une seule exécution d'un calcul et qu'après cette exécution un ordinateur parmi dix fournissait un résultat différent, comment être sûr que le problème provient de cet ordinateur et non pas des neuf autres ? Par conséquent, si la contrainte de reproduction n'est pas satisfaite par une machine il est à la fois impossible d'identifier les bugs du programme et les problèmes physiques de ses composants *via* des calculs exécutés en parallèle.

Supposons ensuite que la contrainte de configuration de soit pas satisfaite par une machine. Si cette contrainte n'est pas satisfaite, cela signifie que l'utilisateur ne peut pas choisir la donnée en entrée qui sera fournie à la machine. Plus précisément, deux cas peuvent avoir lieu. Dans un premier cas, l'utilisateur ne peut pas choisir les données en entrée mais peut avor accès à ces données. Dans un second cas, l'utilisateur ne peut pas choisir les données en entrée car elles sont inaccessibles. Puisque l'utilisateur n'a aucune emprise sur les conditions initiales du calcul quelque soit le cas auquel il peut être soumis, son incapacité à choisir la donnée entrée d'un calcul implique aussi une incapacité à reproduire le calcul - le calcul pourrait être reproduit mais cette reproduction serait indépendante de la volonté de l'utilisateur. D'après le paragraphe précédent, l'incapacité à reproduire le calcul rend par conséquent impossible à la fois l'identification de bugs et les tests parallèles.

Le cas de la contrainte de lisibilité découle du cas de la contrainte de configuration car si la contrainte de lisibilité n'est pas satisfaite alors il en va de même pour la configuration. En particulier, si les données en entrée ne sont pas lisibles cela signifie que l'utilisateur ne peut pas différencier certaines de ces données et donc qu'il ne peut pas les choisir puisqu'elles sont indiscernables. Même si les données en sortie de la machine étaient lisibles, l'utilisateur ne pourrait tout de même pas avoir accès aux données en entrée et les choisir correctement. En effet, pour accéder à ces données en entrée la stratégie suivante pourrait être mise au point : puisque l'utilisateur a configuré l'ordinateur pour calculer une fonction f, il pourrait retrouver, à partir du résultat, la donnée en entrée correspondante. Cependant, une telle

stratégie est facilement mise à mal car si la machine fournit deux fois un résultat lisible qui est identique cela ne voudra pas dire pour autant qu'elle aura exécuté deux fois le même calcul. Plus précisément, certaines fonctions telles que la fonction produit définie par $f(n, m) = n \times m$ où $\forall\ n, m \in \mathbb{N}$, ne sont pas injectives car deux données en entrée distinctes peuvent conduire au même résultat - par exemple $f(1, 0) = 0$ et $f(9, 0) = 0$). Autrement dit, l'utilisateur ne saura pas si le résultat fourni par la machine est le résultat qu'il voulait obtenir. Au final, l'absence de lisibilité conduit à la non satisfaction de la contrainte de configuration.

Pour les contraintes de définissabilité et d'accès, les choses sont plus simples. D'une part, si l'utilisateur ne peut pas définir la fonction avant que la machine commence à la calculer cela veut dire (1) que la machine n'a pas été programmée par l'utilisateur ; ou (2) qu'elle n'est pas programmable. Dans les deux cas, aucune technique liée à la vérification des programmes ne peut être utilisée. D'autre part, si l'utilisateur ne peut pas avoir accès aux résultats fournis par la machine, aucun de ces résultats ne peut être vérifié.

Enfin, la contrainte de fiabilité est nécessaire pour effectuer la vérification des preuves formelles établissant la correction du programme d'une machine. J'ai expliqué précédemment que les preuves conçues par les mathématiciens dans le but de vérifier les programmes d'un ordinateur étaient longues et fastidieuses, ce qui rendait de fait nécessaire l'utilisation d'ordinateurs. Maintenant, si les ordinateurs chargés de vérifier les preuves ne sont pas fiables du point de vue de leurs composants, l'utilisation d'ordinateurs pour la vérification de preuves n'a plus d'intérêt. Cette vérification revient alors aux mathématiciens dont le travail implique des problèmes tels que des taux d'erreurs élévés et des temps de vérification importants.

En résumé, les contraintes inscrites au sein de la condition d'utilisabilité doivent être satisfaites par une machine physiquement construite pour que l'on puisse vérifier partiellement ses résultats à partir des techniques de vérification précédentes. Autrement dit, si la machine ne satisfait pas ces contraintes il est impossible d'affirmer qu'elle calcule une fonction f.

Pour récapituler, j'ai défendu tout au long de cette seconde section deux thèses. J'ai défendu d'une part que même si l'on ne pouvait pas vérifier

parfaitement les valeurs des fonctions calculées par un ordinateur, il était tout de même possible de les vérifier partiellement et d'affirmer qu'un ordinateur calcule des fonctions Turing-calculables. J'ai expliqué d'autre part que la vérification partielle ne pouvait avoir lieu que si la machine était utilisable, c'est-à-dire uniquement si elle satisfaisait l'ensemble des contraintes inscrites au sein de la condition d'utilisabilité.

Puisque que l'on dispose à présent d'un critère - l'utilisabilité - pour déterminer si une machine calcule des fonctions Turing-calculables, on peut tenter de répondre à la question centrale de ce chapitre : est-il possible de vérifier partiellement les résultats fournis par une HM physiquement construite ? Si la réponse à cette question est positive, cela signifierait que l'on dispose d'un critère permettant de démontrer qu'un dispositif hyper-calcule. Cette démonstration attesterait que l'on a construit une HM et donc que la TPHC a bien été démontrée. Dans la dernière section de ce chapitre, je vais mettre à l'épreuve les HM actuellement proposées afin de déterminer si elles seraient utilisables dans l'hypothèse où elles étaient physiquement construites.

3 La thèse physique de l'hyper-calcul peut-elle être démontrée ?

Dans cette dernière section, mon but est d'évaluer si la TPHC pourrait être démontrée. Pour ce faire, je vais supposer que les principales HM qui ont été proposées sont physiquement construites, puis je vais déterminer si elles sont capables de satisfaire la condition d'utilisabilité. Je commence par une analyse préliminaire du problème de la vérification appliqué à l'hyper-calcul puis je mets à l'épreuve les HM qui ont été proposées en vue de démontrer la TPHC.

3.1 Préliminaires

Avant d'analyser une à une les HM, je souhaite tout d'abord expliquer pourquoi les contraintes inscrites au sein de la condition d'utilisabilité sont

pertinentes pour résoudre le problème de la vérification appliqué à l'hyper-calcul.

Copeland a été l'un des premiers a défendre explicitement que la vérification des valeurs de fonctions non Turing-calculables pouvait être un réel problème pour l'hyper-calcul :

> Il y a un problème épistémologique lié à l'hyper-calcul. Supposons que le génie de Laplace dise 'Voici une boîte noire capable de résoudre le problème de l'arrêt' (le problème est soulevé quelle que soit la fonction non calculable par machine de Turing qui est considérée). Tapez n'importe quel entier x et la boîte fournira la valeur de la fonction arrêt $H(x)$ correspondante. Puisqu'il n'y a pas de méthode systématique pour calculer les valeurs de la fonction arrêt, vous n'avez pas de moyen de vérifier si la machine produit les réponses correctes ou non. Même simuler la machine de Turing en question ne vous aidera pas en général : quelle que soit la durée durant laquelle vous regarderez la simulation, vous ne pourrez pas inférer que la machine ne s'arrêtera pas à partir du fait qu'elle ne s'est pas encore arrêtée [Copeland, 2002b, pp. 490-491].

D'après Copeland, si on fait l'hypothèse selon laquelle une machine physiquement construite est une HM, il est impossible de montrer que cette hypothèse est vraie. La raison en est que la démonstration de cette hypothèse repose sur notre capacité à vérifier que la machine calcule au moins une fonction non Turing-calculable. Or une telle vérification est impossible car on ne détient pas de méthode systématique - de procédure effective - permettant, à partir d'une donnée en entrée, de suivre chacune des étapes du calcul et d'arriver en un temps fini au résultat. Autrement dit, toute tentative de vérification des propriétés hyper-calculatoires d'une HM est vouée à l'échec.

Copeland a raison de conclure à partir de son expérience de pensée qu'il est impossible de vérifier les résultats fournis par une HM. Toutefois, cette conclusion n'est valable que pour deux raisons :

1. La première raison est que l'expérience énoncée plus haut suppose de

pouvoir vérifier les résultats indépendamment de tout contact physique avec la machine. Il est par exemple impossible dans l'expérience du génie de Laplace d'ouvrir la boîte noire pour comprendre le fonctionnement interne de la machine. Cependant, le contact physique avec le dispositif - que se soit un ordinateur ou une HM - est fondamental afin de vérifier ses résultats. En effet, j'ai défendu plus haut que c'est l'exécution de tests théoriques et empiriques sur le fonctionnement des ordinateurs et non un suivi étape par étape qui permettait d'affirmer qu'une fonction donnée est calculée par un ordinateur. Sans le contact physique avec l'ordinateur il serait donc impossible d'affirmer qu'il calcule. Il en va de même pour une HM : le contact physique avec le dispositif est nécessaire pour effectuer des tests théoriques et empiriques dans le but d'affirmer qu'elle hyper-calcule.

2. La seconde raison est que Copeland considère uniquement une vérification totale par opposition à une vérification partielle. Cette vérification totale est en particulier impossible dans le cas des HM puisque l'on ne peut pas suivre chaque étape du calcul d'une fonction non Turing-calculable. En revanche, il pourrait être possible de vérifier partiellement les résultats fournis par une HM. Autrement dit, il pourrait être possible d'augmenter la confiance que l'on attribue aux calculs exécutés par l'HM jusqu'à un niveau suffisant permettant d'affirmer qu'elle hyper-calcule.

Le problème de la vérification tel qu'il est énoncé par Copeland est ainsi un réel problème pour l'hyper-calcul car (1) la vérification est ménée indépendamment de tout contact physique avec l'HM ; et (2) la vérification est considérée comme devant être totale. En revanche, si l'on permettait à l'examinateur d'avoir un contact physique avec l'HM afin qu'il puisse effectuer des tests théoriques et empiriques dans le but de vérifier partiellement ses résultats, il se pourrait que le problème de la vérification appliqué à l'hyper-calcul soit résoluble. En particulier, ces tests permettraient d'affirmer qu'une HM calcule une fonction non Turing-calculable alors même qu'une vérification totale est impossible. Toutefois, comment établir que ces tests peuvent être effectués sur une HM ? La réponse à cette question est la

même que pour les ordinateurs : les tests de vérification partielle peuvent être effectués sur une HM si cette HM est utilisable, c'est-à-dire si elle satisfait les contraintes inscrites au sein de la condition d'utilisabilité. En d'autres termes, l'utilisabilité d'une HM est une condition nécessaire et suffisante à la vérification partielle de ses résultats.

Il existe en outre une différence importante entre la démarche que j'ai entreprise dans la section précédente pour montrer que les ordinateurs sont utilisables et celle que je vais maintenant effectuer pour évaluer si les HM sont utilisables. Cette différence est due à la nature des définitions des notions d'ordinateurs et d'HM. Dans le cas des ordinateurs, il existe une définition générale de la notion d'ordinateur ou du moins une définition qui comporte un ensemble de caractéristiques propres aux ordinateurs, et cela malgré leur grande pluralité. Ces caractéristiques telles que leur universalité - propriété selon laquelle les ordinateurs calculent les mêmes fonctions que la MT - sont en effet acceptées par les scientifiques, et cette acceptation a permis l'utilisation d'une définition générale de la notion d'ordinateur. Pour les HM en revanche, il n'existe pour l'instant aucune définition générale comportant un ensemble de caractéristiques qui seraient partagées par toutes les HM. La raison principale en est que les HM qui ont été proposées jusqu'ici possèdent des fonctionnements radicalement différents pouvant être fondés sur l'accélération du calcul ou sur l'aléatoire quantique.

La conséquence de cette différence entre les définitions des notions d'ordinateurs et d'HM en ce qui concerne la résolution du problème de la vérification est la suivante. Contrairement aux ordinateurs pour lesquels j'ai pu défendre sans les différencier qu'ils satisfaisaient les contraintes d'utilisabilité, je vais devoir dans ce qui suit analyser une par une les principales HM qui ont été proposées dans le but de déterminer si elles satisfont les contraintes d'utilisabilité.

3.2 L'hyper-calcul est-il utilisable ?

Serait-il possible de démontrer la TPHC en vérifiant partiellement les résultats fournis par une HM physiquement construite ? Autrement dit, est-

ce qu'une HM physiquement construite pourrait satisfaire les contraintes d'utilisabilité ? Je commence par répondre négativement à cette question pour quatre types d'HM : les machines à essais et erreurs, les HM manipulant des nombres réels, les HM quantiques à processus adiabatiques et les HM relativistes. J'étudie ensuite séparément le cas des machines à oracles aléatoires (OMA) qui sont les HM les plus prometteuses en vue d'une possible démonstration de la TPHC.

3.2.1 Machines à essais et erreurs

Schématiquement, une machine à essais et erreurs (MEE) est une HM qui est toujours capable de fournir après un certain intervalle de temps la valeur en sortie d'une fonction f Turing-calculable ou non [Gold, 1965], [Putnam, 1965].

La particularité de la MEE réside dans le fait qu'elle peut fournir des réponses fausses avant de fournir la réponse correcte. Supposons que la MEE calcule une fonction non Turing-calculable $f : \mathbb{N} \rightarrow \{0, 1\}$ et considérons une donnée en entrée n que l'on fournit à la machine. D'après son fonctionnement, la MEE fournira bien la valeur correcte de $f(n)$, disons 1, mais pourra alterner un nombre arbitraire de fois entre 0 et 1 avant d'inscrire définitivement 1.

D'un point de vue pratique, le problème est que la MEE n'est pas utilisable car il est impossible pour un observateur fini de déterminer si la dernière réponse fournie par la MEE est une réponse correcte. Dans le cas contraire, cela reviendrait à résoudre le problème de l'arrêt propre aux MT. Plus précisément, la MEE n'est pas utilisable car elle ne satisfait pas la contrainte de lisibilité. Même s'il peut sembler que la donnée en sortie fournie par la machine soit lisible, au sens où l'observateur peut déterminer si cette donnée est 0 ou 1, elle est au contraire illisible. En effet, une donnée en sortie est par définition lisible si l'observateur peut lire cette donnée au degré d'approximation souhaité. Or puisque le degré d'approximation souhaité est la dernière valeur fournie par la MEE - autrement dit la réponse correcte - et que l'observateur ne sait pas s'il est confronté à cette valeur, la donnée en sortie n'est pas lisible pour l'observateur.

3.2.2 Hyper-machines manipulant des nombres réels

Tout au long de cette thèse, j'ai présenté à plusieurs reprises des HM ayant la particularité de manipuler des nombres réels. Les machines BSS [Blum et al., 1926], les machines accumulatrices [Copeland, 1997] et les X-machines [Stannett, 1990] ont été introduites au chapitre 1 ; tandis que les réseaux de neurones artificiels ont été étudiés au chapitre 2 [Siegelmann, 1999].

Même si ces quatre HM peuvent calculer des fonctions non Turing-calculables à l'aide de nombres réels, elles ne seraient pas utilisables dans l'hypothèse où elles étaient physiquement construites. Cela peut paraître surprenant du fait que les ordinateurs analogiques manipulant eux aussi des nombres réels étaient autrefois utilisés avant l'avènement des ordinateurs digitaux [Rubel, 1993], [Mills, 2008]. Il existe cependant une différence fondamentale entre les ordinateurs analogiques et les quatre HM présentées ci-dessus. Tandis que les ordinateurs analogiques manipulent des variables qui ne sont pas des valeurs exactes mais des approximations de nombres réels, les HM citées plus-haut reposent quant à elles sur l'utilisation explicite de variables pouvant prendre des valeurs exactes de nombres réels. En particulier, lorsque qu'une variable prend la valeur d'un nombre non Turing-calculable, cette variable n'est pas une approximation de la valeur d'un nombre réel puisque si tel était le cas, cette valeur serait finie et donc Turing-calculable.

C'est justement parce que ces quatre hyper-machines manipulent des valeurs exactes de nombres réels qu'elles seraient inutilisables si on les construisaient physiquement. Elles seraient inutilisables car leurs données en entrée seraient illisibles : l'utilisateur serait en effet incapable de fournir à la machine des données en entrée au degré d'approximation désiré. Cette incapacité peut être expliquée de la manière suivante. D'une part, puisque la donnée doit prendre pour valeur la valeur exacte d'un nombre réel, l'utilisateur doit pouvoir fournir à la machine un nombre dont les décimales sont infinies, ce qui est impossible à cause des capacités finies de l'utilisateur. D'autre part, la donnée fournie à la machine par l'utilisateur doit être non Turing-calculable ce qui conduit immédiatement à la contradiction suivante : si l'utilisateur

produit un nombre non Turing-calculable, il y aurait de grandes chances que le processus de calcul qui lui a permis de produire ce nombre soit lui-même Turing-calculable.

3.2.3 Hyper-machines quantiques à processus adiabatiques

Les machines quantiques à processus adiabatiques sont proposées par Kieu - *cf.* chapitre 3. Les travaux de Kieu sur ce type d'HM forment un ensemble de dix articles dont quatre ont été publiés dans des journaux internationaux [Kieu, 2002], [Kieu, 2003], [Kieu, 2004], [Kieu, 2005].

Pour rappel, une HM quantique à processus adiabatiques ou plus simplement une HM adiabatique utilise un algorithme quantique qui repose sur le *théorème adiabatique* [Kato, 1950]. Cet algorithme a pour principal avantage de pouvoir encoder, au sein d'un certain état fondamental, une instance particulière d'un problème indécidable donné. Le système commence son évolution à partir de l'état quantique d'un hamiltonien aisé à construire, qui est le plus souvent l'état du système possédant le plus bas niveau d'énergie. Le système évolue ensuite très lentement vers un hamiltonien que l'on souhaite atteindre. D'après le théorème adiabatique et un ensemble de conditions précises, le résultat du système prendra la forme d'un autre état fondamental qui encode la solution du problème indécidable qui a été codé lors de la phase initiale de l'algorithme.

A supposer que l'on dispose d'une HM adiabatique physiquement construite, cette dernière ne serait pas utilisable. Plus précisément, cette HM ne satisfait pas les contraintes de reproduction et de lisibilité. Schématiquement, l'impossibilité pour l'HM de satisfaire ces deux contraintes a pour origine le caractère probabiliste de son mode de fonctionnement. En effet, le résultat final de l'algorithme quantique utilisé par l'HM adiabatique dépend de l'intervalle de temps consacré à l'évolution du système : plus longtemps le système évolue, plus la probabilité d'obtenir le bon résultat augmente.

Plus précisément, l'HM adiabatique ne satisfait pas la contrainte de reproduction puisqu'il est clair que des évolutions successives du système conduisent à des résultats différents. De même, l'HM adiabatique ne satisfait pas la

contrainte de lisibilité pour la raison suivante : on ne sait pas si le résultat produit par l'HM est le résultat recherché. En particulier, à quel moment doit-on mesurer le système afin d'obtenir le résultat du calcul ? Il est en effet impossible de déterminer où en est l'évolution puisque toute mesure autre que la mesure finale pertubera le système du fait que son évolution doit se faire sans contact avec l'extérieur - c'est le principe même d'une évolution dite adiabatique. En outre, puisque la probabilité d'atteindre l'état désiré est proportionnelle au temps que met le système à évoluer, comment déterminer l'intervalle de temps qui doit s'écouler entre le début de l'évolution et la mesure ? Autrement dit, à quel moment doit-on prendre la décision de mesurer le système puisque le fait de ne pas le mesurer augmente simultanément la probabilité d'obtenir le bon résultat ?

En conclusion, même si l'HM de Kieu était physiquement construite elle ne serait pas utilisable.

3.2.4 Hyper-machines relativistes

Les HM relativistes exploitent les propriétés d'un type particulier d'espace-temps nommés *espace-temps de Malament-Hogarth*. Schématiquement, les espace-temps de Malament-Hogarth sont composés de régions contenant une trajectoire infinie λ qui peut être contournée par une trajectoire finie g. Autrement dit, λ et g ont une origine commune, appelée une *bifurcation*, mais il existe un point de l'espace-temps p sur g tel que λ est entièrement contenue dans le passé chronologique de p.

Une HM relativiste est composée de deux ordinateurs modernes pouvant exécuter, en plus des opérations propres aux MT, les opérations suivantes : (1) se positionner à une bifurcation où débutent λ ou g, (2) envoyer un des deux ordinateurs le long de λ tandis que l'autre reste sur g, (3) atteindre le point p à partir d'où g a contourné λ, (4) envoyer un signal de λ à g, et (5) recevoir un signal provenant de λ. Ces opérations permettent en particulier de définir une procédure qui permet aux HM relativistes de résoudre le problème de l'arrêt propre aux MT - *cf.* chapitre 3.

Supposons à présent que l'on dispose d'une HM relativiste physiquement

construite et posons la question suivante : cette HM est-elle utilisable ? La réponse que je vais donner à cette question est similaire à celle que propose Piccinini [Piccinini, 2011, pp. 757-760]. Cette réponse consiste à dire qu'il est *probable* mais non certain que les HM relativistes ne soient pas utilisables.

Les raisons qui empêchent d'affirmer que les HM relativistes sont utilisables proviennent d'une part de l'affirmation selon laquelle les contraintes de lisibilité et de fiabilité ne sont pas satisfaites. Il semblerait en effet d'après certains chercheurs spécialistes du domaine que ces contraintes ne soient pas satisfaites par les HM relativistes, même si aucune preuve ne vient appuyer cette dernière affirmation. Elles proviennent d'autre part du nombre important d'hypothèses devant être réalisées pour satisfaire les contraintes de reproduction et de configuration. Commençons par détailler ce dernier point.

De la manière dont les HM relativistes ont été initialement définies par Hogarth, elles ne satisfont pas la contrainte de reproduction car elles ne peuvent avoir accès qu'à une unique région de l'espace-temps [Hogarth, 1994]. Plus précisément, si une HM relativiste n'a accès qu'a une seule région de l'espace-temps alors elle ne peut effectuer qu'un seul calcul. Pour des raisons qui ne sont pas liées à la reproduction des calculs, Hogarth a néanmoins supposé que les espace-temps de Malament-Hogarth contenaient une grande quantité de régions. En particulier, si l'on permet à ces espace-temps de contenir un grand nombre de régions, il est possible de montrer que les HM relativistes satisfont la contrainte de reproduction. Si ces HM ont en effet accès a plusieurs régions différentes les unes après les autres, chacune de ces régions peut être exploitée afin d'effectuer un calcul différent. Cela permet ainsi aux HM relativistes de calculer plusieurs valeurs d'une même fonction et de satisfaire du même coup la contrainte de reproduction.

La satisfaction de la contrainte de configuration demande elle aussi quelques concessions. Pour rappel, une machine satisfait la contrainte de configuration si l'utilisateur peut lui fournir pour données en entrée les différents arguments de la fonction qui est calculée. Lorsque l'utilisateur souhaite commencer un nouveau calcul, la machine doit aussi pouvoir être réinitialisée à son état initial pour qu'un nouvel argument - qui peut être identique au précédent - puisse lui être fourni. Malgré la complexité de cette définition, les HM relati-

vistes satisfont la contrainte de configuration. Cependant, cette satisfaction nécessite de faire deux concessions. D'une part, on est obligé de supposer ici aussi que l'espace-temps de Malament-Hogarth contient un grand nombre de régions pouvant être exploitées afin de préparer son état initial au calcul d'un nouvel argument. D'autre part, même si les HM relativistes sont configurables au sens où elles peuvent calculer plusieurs valeurs d'une fonction, il est en revanche impossible de la reconfigurer pour qu'elles calculent différentes fonctions. L'ordinateur perd en particulier certains de ses composants lorsqu'il est lancé le long de λ, rendant de fait impossible tout nouveau lancement.

Malgré le nombre important d'hypothèses nécessaires pour satisfaire les contraintes de reproduction et de configuration, les HM relativistes sont pour l'instant utilisables. Cependant, leur incapacité à satisfaire les contraintes de lisibilité et de fiabilité les rendent en définitive inutilisables. La cause de cette incapacité a pour origine la difficulté de décoder le signal qui provient de λ et qui contient le résultat du calcul. Il est en effet très difficile de différencier le signal contenant le résultat du calcul avec les autres signaux pouvant provenir de l'espace. En outre, le signal peut être soumis à un phénomène dit *d'amplification*. Cette amplification a deux conséquences négatives différentes suivant que le signal contenant le résultat est infini ou fini. Dans le cas où le signal est infini, Earman et Norton soutiennent que l'ordinateur ayant pour instruction de réceptionner le signal provenant du second ordinateur lancé le long de λ sera détruit [Earman and Norton, 1993]. Dans le cas où le signal est fini, Etesi et Németi affirment que les forces gravitationnelles auront pour effet de diminuer la longueur du signal, et ce jusqu'à la fin du calcul où la longueur du signal convergera vers zéro [Etesi and Németi, 2002]. Pour contourner ce problème, Etesi et Németi admettent que le décodage du signal nécessite la possibilité d'exécuter des mesures avec une précision arbitrairement petite, possibilité qui semble être en désaccord avec les contraintes de la mécanique quantique. Il existe toutefois une autre solution capable de résoudre le problème de la réception. Schématiquement, cette solution consiste à envoyer un messager de λ vers g : bien que le messager ne puisse pas atteindre g, il pourrait se positionner assez près de g pour transmettre

le signal sous une forme lisible et cela sans dégrader l'ordinateur resté sur g [Németi and Dàvid, 2006], [Andréka and Németi, 2009].

Même s'il existe en théorie une solution possible au problème de la réception, cette solution repose sur un ensemble conséquent d'hypothèses qui, d'après moi, interdit la conclusion selon laquelle les HM relativistes sont utilisables. Autrement dit, au même titre que les machines à essais et erreurs, les HM manipulant des nombres réels et les HM à processus adiabatiques, les HM relativistes échouent à satisfaire les contraintes d'utilisabilité.

Cet échec a pour principale conséquence d'élever certains doutes quant à la possibilité de démontrer la TPHC. Le résultat selon lequel les quatre HM étudiées ci-dessus ne sont pas utilisables montre en effet que la construction physique d'une de ces HM ne serait pas suffisante pour démontrer la TPHC. Plus précisément, si l'une d'entre elles venait à être physiquement construite, il serait impossible de vérifier partiellement que ses résultats correspondent aux valeurs d'une fonction non Turing-calculable.

Cet échec ne remet cependant pas définitivement en cause la démonstration de la TPHC. J'ai en particulier défendu au chapitre 3 que la proposition la plus prometteuse en vue de démontrer cette thèse est fondée sur l'aléatoire provenant à la fois de la théorie quantique [Calude, 2005] et de la théorie de la radioactivité [Stannett, 2003]. Ces propositions sont les plus prometteuses car leur construction physique est plus réaliste que celle des HM à processus adiabatiques ou des HM relativistes [3]. Pour apporter une réponse précise à la question d'une possible démonstration de la TPHC, je consacre ainsi le reste de ce chapitre aux HM à oracles aléatoires (OMA).

3.3 Hyper-machines à oracles aléatoires

Le chapitre 3 a eu pour but de défendre la thèse selon laquelle les OMA doivent être considérées comme la meilleure proposition de démonstration de la TPHC. Plus précisément, mon principal argument était que les deux OMA qui sont actuellement proposées ont de grandes chances d'être physiquement

3. Consulter en particulier [Hawking, 1975], [Earman and Norton, 1993], [Hodges, 2005], [Hagar and Korolev, 2006]

construites. En effet, la première de ces deux HM est fondée sur l'aléatoire quantique et utilise un processus optique qui a été implémenté par l'entreprise *ID Quantique*, tandis que la seconde est fondée sur la désintégration radioactive et dispose d'un protocole expérimental attendant d'être mis à l'épreuve.

Cependant, la construction d'une OMA n'implique pas obligatoirement son utilisabilité et il se pourrait que l'on parvienne à construire une telle machine sans être capable d'affirmer qu'elle hyper-calcule. Dans ce dernier cas, il serait impossible de vérifier que la machine est bien une HM et donc que la TPHC a été démontrée.

Voici justement le but que je souhaite atteindre ici : apporter une réponse quant à la possibilité de vérifier partiellement que les OMA hyper-calculent. Pour ce faire, je vais procéder en deux temps. Je vais commencer dans un premier temps par montrer que ces HM ne sont pas utilisables au sens où elles ne satisfont pas certaines des contraintes incluses dans la condition d'utilisabilité d'une machine. Je proposerai et j'évaluerai ensuite dans un second temps l'efficacité de certaines méthodes heuristiques qui pourraient servir de pistes dans le but de vérifier les résultats fournis par les OMA.

3.3.1 Utilisation

Pour rappel, il existe deux tentatives de construction d'une OMA : celle de Stannett [Stannett, 2003] et celle de Calude [Calude, 2005]. Ces deux tentatives, même si elles ne sont pas fondées sur la même théorie physique, ont pour point commun la stratégie de produire une suite de nombres aléatoires qui représente les valeurs d'une fonction non Turing-calculable - *cf.* chapitre 3.

Passons à présent à la question de l'utilisabilité des o-machines. Sur cette question, mon point de vue général est que les o-machines ne sont pas utilisables. Dans le détail, elles ne satisfont pas les contraintes d'identification, de reproduction et d'accès.

Identification. Tout d'abord, une OMA ne satisfait pas la contrainte d'identification car il n'y a aucun moyen de définir la fonction $f : \mathbb{N}^k \to \{0, 1\}$ dont

les valeurs sont produites par le processus aléatoire.

Plus précisément, une fonction mathématique peut être définie de deux manières différentes : en extension ou en compréhension. Informellement, une fonction f à n arguments est définie en *extension* si elle est définie à partir de l'ensemble des couples $(x_1, \ldots, x_n, m)$ qui la caractérisent. Par exemple, la fonction unaire f ayant pour domaine $D_f = \{0, 1\}$ et étant caractérisée par les couples $\{(0, 1), (1, 0)\}$ est définie en extension. Une fonction f à n arguments est définie en *compréhension* si elle est définie sans inscrire l'ensemble des couples $(x_1, \ldots, x_n, m)$ qui la caractérise. Citons par exemple les fonctions binaire *somme* et unaire *racine carré* respectivement définies sur $\mathbb{N}$ par $f(x, y) = x + y$ et $f(x) = \sqrt{x}$. De telles définitions par compréhension peuvent être considérées comme des *opérations* au sens où elles fournissent chacune une procédure pour calculer les valeurs de la fonction qui est définie [Russell et al., 1994, p. 625].

Que l'on souhaite obtenir une définition en extension ou en compréhension de la fonction f calculée par l'OMA, obtenir cette définition est impossible. Il est en effet impossible de définir en extension la fonction qui est calculée par une OMA, et cela pour une raison très simple : puisque la suite de nombres qui est produite par l'OMA doit par définition être infinie pour être aléatoire, il est impossible d'énumerer l'ensemble infini des couples $(x_1, \ldots, x_n, m)$ qui la caractérisent. On peut de même montrer à partir d'un raisonnement par l'absurde qu'il est impossible de définir cette fonction en compréhension. Supposons que l'on puisse définir en compréhension la fonction calculée par l'OMA. Puisque le processus exécuté par l'OMA est aléatoire, la définition en compréhension de la fonction n'a pu se faire qu'à partir de la suite de nombres fournie par l'OMA. Cela signifie plus précisément que l'on a réussi à trouver, à partir d'une suite finie de nombres aléatoires, un moyen de produire cette suite. On dispose par conséquent d'une règle ou d'une *opération* pour reprendre le vocabulaire de Russell permettant de produire un à un les nombres qui composent la suite produite par l'OMA. Toutefois, le fait de disposer d'une telle règle est contradictoire avec l'hypothèse selon laquelle le processus est aléatoire.

Reproduction. Même si l'on peut réitérer le processus aléatoire, c'est-à-dire produire à nouveau une SNA à partir du processus, il est néanmoins impossible de produire une SNA qui serait identique à la précédente. En effet, puisque le processus est aléatoire on ne dispose d'aucun moyen permettant de produire une SNA particulière ; on ne dispose en d'autres termes d'aucune opération capable de produire les valeurs d'une fonction non Turing-calculable donnée. Il suit qu'il est impossible de reproduire le processus aléatoire qui a permis de produire une SNA particulière.

Plus formellement, quelle est la probabilité que la seconde suite soit identique à la précédente ? Tout d'abord, la suite est par définition équilibrée et donc régie par la loi des grands nombres : la proportion de 0 et de 1 doit donc être équivalente au sein de la SNA lorsqu'une grande quantité de nombres est produite - chaque symbole à donc une probabilité d'apparaître égale à 0,5. La loi des grands nombres permet en particulier de calculer la probabilité de produire une suite particulière qui aurait été produite lors de la précédente utilisation de l'OMA. Soit S la SNA définie par

$$S = (x_0, x_1, x_2, x_3, \ldots, x_n, \ldots)$$

La probabilité que les n premiers nombres de la suite S' nouvellement produite soient identiques à ceux de la suite S est égale à

$$\frac{1}{2^n}$$

Et puisque la suite S est composée d'une infinité de nombres, la probabilité que $S = S'$ est égale à

$$\lim_{n\to\infty} \frac{1}{2^n} = 0$$

Il est donc impossible qu'une SNA soit produite par une OMA deux fois de suite.

Accès. Pour Piccinini, une OMA ne satisfait pas la contrainte de configuration [Piccinini, 2011, p. 751]. L'auteur explique qu'une OMA n'est pas

configurable car l'utilisateur ne peut pas choisir la valeur de f qu'il souhaite calculer :

> Mais l'utilisateur d'un processus aléatoire P ne peut pas sélectionner la valeur de f qu'il souhaite que P produise. S'il souhaite obtenir la n-ième valeur de f, tout ce qu'il doit faire consiste à exécuter P et à attendre jusqu'à ce que la n-ième valeur soit générée. Et tant pis si cela ne se produit pas avant un million d'années [Piccinini, 2011, p. 751].

Je ne suis pas tout à fait en accord avec Piccinini car le problème que soulève l'OMA est selon moi davantage lié à la contrainte d'accès qu'à la contrainte de configuration. Plus précisément, le fait que l'utilisateur soit obligé d'attendre que l'OMA produise $n-1$ valeurs avant d'atteindre la valeur n souhaitée n'est pas un problème en soi. Les programmeurs informatiques utilisent en effet très fréquemment des *algorithmes récursifs* pour permettre aux ordinateurs de calculer certaines fonctions, algorithmes qui nécessitent le calcul des $n-1$ premières valeurs pour parvenir à la valeur n. Schématiquement, un algorithme - respectivement une fonction - est récursif - respectivement récursive - lorsqu'il ou elle contient un appel à lui-même ou à elle-même. Par exemple, la fonction factorielle définie sur $\mathbb{N}$ par $n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$ est récursive [4].

Supposons à présent que l'on implémente un algorithme récursif pour calculer la suite de Fibonacci $\{F_n\}_{n\geq 1}$ telle que $F_1 = F_2 = 1$ et $F_{n+1} = F_n + F_{n-1}$ - intuitivement la suite de nombres dans laquelle tout nombre est égal à la somme des deux précédents. Supposons par ailleurs qu'un utilisateur souhaite calculer F_{100}. Par définition de l'algorithme récursif servant à calculer F, l'utilisateur doit attendre que l'ordinateur calcule les valeurs antérieures à celle de F_{100}. Cette attente empêche-t-elle pour autant l'ordinateur d'être utilisable ? Non car la suite de Fibonacci a de nombreuses applications pratiques. Par exemple, l'analyse technique en finance utilise un outil appelé

4. L'expression *fonction récursive* utilisée ici ne doit pas être confondue avec la notion de fonction récursive introduite par Kleene qui renvoie aux fonctions calculables par MT [Kleene, 1936]. Il se peut toutefois qu'une fonction soit récursive à la fois parce qu'elle contient un appel à elle-même et parce qu'elle est calculable par une MT. La fonction factorielle qui vient d'être présentée en est un exemple.

retracement de Fibonacci qui permet de prédire les mouvements boursiers en fonction de ratios ou de seuils qui font référence à la suite de Fibonacci. Par conséquent, une machine qui produit les valeurs antérieures à la valeur que l'on souhaite calculer ne peut pas être considérée comme étant inutilisable.

On pourrait néanmoins objecter que le calcul d'une fonction récursive ne nécessite pas toujours le calcul des valeurs antérieures à celle que l'on souhaite calculer. Par exemple, le calcul du n-ième terme d'une fonction F de la forme $F_{n+1} = F_n + r$ avec $n, r \in \mathbb{N}$ peut se faire directement à partir de la formule $F_n = F_0 + (n-1) \times r$. En particulier, si ce type de *raccourci* existait pour la suite de Fibonacci et plus généralement pour toute suite récursive, la nécessité de calculer les valeurs antérieures à la valeur que l'on souhaite calculer serait endémique aux o-machines.

Toutefois, le problème de déterminer s'il existe des *raccourcis* permettant de calculer plus rapidement les valeurs de fonctions récursives n'est pas encore résolu. Les chercheurs travaillant sur ce problème tentent actuellement de définir rigoureusement ce qu'est un raccourci permettant de calculer plus rapidement. Delahaye et Zwirn ont notamment proposé une définition formelle de *l'irréductibilité computationnelle*, notion traduisant le fait d'être incapable de calculer $f(n)$ sans avoir à calculer $f(i)$ pour tout $i < n$ [Zwirn and Delahaye, 2013]. L'existence de fonctions irréductibles serait ainsi un contre-exemple à l'argument de Piccinni, puisque, dans le cas où l'on souhaiterait obtenir une valeur n, la nécessité d'attendre que les $n - 1$ premières valeurs soient calculées serait une propriété intrinsèque de certains calculs.

Selon moi, le problème que soulèvent les o-machines ne provient pas de la contrainte de configuration mais de la contrainte d'accès. La fin du passage écrit par Piccinini insiste très justement sur ce point : si la production du n-ième symbole de la SNA nécessite un million d'années, il est impossible d'accéder aux résultats produits par l'OMA. Autrement dit, le temps que met le processus aléatoire à produire une SNA est un obstacle à la satisfaction de la contrainte d'accès. Il cependant très subtil de comprendre pourquoi l'impossibilité d'accéder à certaines valeurs calculées par une OMA est différente de l'impossibilité - par manque de ressources physiques - d'accéder à certaines valeurs calculées par un ordinateur moderne. En fait, la seconde impossibi-

lité est différente de la première car elle peut être partiellement surmontée grâce aux avancées technologiques. Par exemple, s'il s'avère impossible par manque de temps de calculer à partir de l'ordinateur M la n-ième valeur d'une fonction, il est néanmoins fort probable que les avancées technologiques permettent de concevoir un ordinateur M' plus rapide que M qui permet de se rapprocher de la valeur n recherchée si ce n'est de l'atteindre. En d'autres termes, même s'il existe toujours des valeurs inaccessibles, la frontière entre résultats accessibles et résultats inacessibles s'éloigne au fur et à mesure que les avancées technologiques progressent. Ce raisonnement est malheureusement impossible à appliquer dans le cas des o-machines, car même si l'on suppose que des avancées technologiques permettront d'augmenter la vitesse de production d'une SNA, on ne pourra pas se rapprocher de la valeur n recherchée. La raison en est que l'augmentation de la vitesse de production d'une SNA demandera de modifier le dispositif utilisé par l'OMA afin de la reproduire plus rapidemment ; or la reproduction d'une SNA est impossible puisque le processus produit aléatoirement une suite à chaque utilisation. Autrement dit, cela signifie que l'on ne se rapprochera pas de la valeur n recherchée car à chaque réitération du processus, cette valeur correspondra à une autre fonction. Une OMA ne satisfait donc pas la containte d'accès.

Ce dernier résultat est un argument puissant allant à l'encontre d'une possible démonstration de la TPHC car il établit que l'OMA ne serait pas utilisable dans le cas où elle serait physiquement construite. Il signifie en particulier que si une OMA est physiquement construite, il serait impossible de vérifier que ses résultats correspondent aux valeurs d'une fonction non Turing-calculable. Un tel résultat est d'autant plus important que les propositions concurrentes de l'OMA, même si elles se révèlent être inutilisables après analyse, ne constituent pas, contrairement à l'OMA, la proposition la plus prometteuse en vue de démontrer la TPHC.

D'après moi, l'impossibilité de vérifier la puissance hyper-calculatoire de l'OMA a pour conséquence majeure de rendre fortement probable la thèse selon laquelle la TPHC est indémontrable. En effet, puisqu'il n'existe aucun moyen permettant de vérifier - même partiellement - que les HM récemment proposées hyper-calculent, aucune preuve de la TPHC ne peut être apportée.

Autrement dit, la TPHC est en quelque sorte condamnée à revêtir le statut de thèse et non d'énoncé scientifique.

L'affirmation selon laquelle la TPHC n'est pas démontrable doit néanmoins être utilisée avec précaution car elle repose sur un point que je n'ai pas expliqué en détail. Ce point concerne la confiance que l'on confère à la vérification partielle, c'est-à-dire la confiance qui est suffisante pour affirmer qu'une machine particulière calcule. J'ai écrit plus haut que la satisfaction des contraintes d'utilisabilité est une condition nécessaire et suffisante pour affirmer avec une forte probabilité qu'une HM hyper-calcule. Cependant, je n'ai pas quantifié cette probabilité. En d'autres termes, il se peut que la probabilité qui découle de la satisfaction des contraintes d'utilisabilité soit *excessive* au sens où l'on requiert une trop grande probabilité afin de pouvoir vérifier les résultats fournis par une HM. Dans ce cas, la conception de méthodes et de techniques pour la vérification ne garantissant pas une probabilité aussi importante que celle garantie par les contraintes d'utilisabilité devient légitime.

3.3.2 Techniques heuristiques pour la vérification

Je propose dans ce qui suit trois techniques heuristiques qui pourraient servir à vérifier les résultats fournis par les HM. J'utilise ici le conditionnel pour deux raisons. La première est que les techniques que je vais proposer font l'objet de travaux en cours [Franchette, 2013] ; je n'exprimerai donc pas d'avis définitifs à leur sujet. La seconde est que ces techniques soulèvent certains problèmes qui mettent en cause leur réalisation.

Vérification à partir du programme. Une première technique consiste à vérifier que le programme de l'HM est correctement spécifié. Une telle technique est courramment utilisée dans le cas des ordinateurs modernes et pourrait être appliquée à l'hyper-calcul. Toutefois, vérifier le programme d'une HM n'est pas toujours possible car les HM ne sont pas toutes programmables.

Selon Loff et Costa, il existe en effet deux types d'HM [Loff and Costa, 2009] : les HM *programmables* et les HM à *oracles*. Les HM programmables

hyper-calculent à partir d'un élément interne - leur programme - et la théorie physique dans laquelle elles sont conçues permet d'exécuter les instructions du programme - par exemple exécuter une boucle infinie en un nombre fini d'étapes. Les HM à processus adiabatiques et les HM relativistes font partie de cette catégorie. Les HM à oracles utilisent en revanche un élément externe - l'oracle - qui provient de la théorie physique au sein de laquelle elles sont conçues.

Dans le cas des HM programmables, une technique fondée sur la vérification du programme permet selon moi de renforcer la confiance que l'on confère aux résultats fournis par ces HM. Considérons les HM relativistes en particulier. Puisqu'une HM relativiste est composée de deux ordinateurs modernes, la vérification de son programme se réduit au problème de vérifier que le programme d'une machine de Turing universelle est correctement implémenté. Les techniques actuelles de vérification du programme d'un ordinateur moderne peuvent donc être appliquées aux HM relativistes.

Une telle technique est en revanche inefficace dans le cas des HM à oracles car ces dernières ne possèdent pas de programme. Cette dernière affirmation doit néanmoins être précisée car il est possible de concevoir un programme pour l'OMA de Calude mais qui ne la rend pas programmable pour autant [5]. En effet, puisque l'OMA est définie comme étant composée d'un ordinateur moderne et d'un oracle - à savoir le composant informatique Quantis - il est possible d'implémenter le programme suivant au sein de l'ordinateur moderne :

1. Faire fonctionner Quantis.
2. Attendre n étapes de calcul. Ces n étapes correspondent à la production des n premières valeurs de la suite aléatoire générée par Quantis.
3. Extraire un symbole m compris entre le premier et le n-ième symbole de la suite.
4. Produire m en sortie. Ce symbole correspond à $f(n)$ où f est une fonction non Turing-calculable.

5. Un grand merci à Jack Copeland et à José-Félix Costa pour ce point important.

Cette suite d'instructions est certes un programme mais elle n'est pas suffisante pour attribuer la *programmabilité* à l'OMA. La programmabilité implique en particulier de pouvoir définir la fonction qui est calculée par le programme, ce qui est impossible dans le cas de l'OMA puisque cette dernière ne satisfait pas la contrainte d'identification. Bien sûr, la fonction f qui est calculée par l'OMA est définie d'un point de vue ensembliste par des couples entrée-sortie ; mais elle n'est pas définie par une spécification de la relation qui existe entre les arguments et les valeurs de f comme cela peut-être le cas avec la fonction arrêt h propre aux MT définie par $h(x, y) = 1$ si et seulement si la MT M_x s'arrête sur l'entrée y, et par $h(x, y) = 0$ autrement.

Vérification à partir de la théorie physique. D'après Shagrir et Pitowsky, l'utilisateur d'une HM n'est pas dans l'incapacité totale d'affirmer qu'un dispositif hyper-calcule car

> Nous pouvons utiliser nos théories physiques pour expliquer quelle fonction est en train d'être calculée. Nos théories physiques permettent d'affirmer qu'un dispositif ayant telle structure ou telles conditions initiales se comportera en fonction d'un ensemble d'équations données [Shagrir and Pitowsky, 2003, p. 26].

Shagrir et Pitowsky expliquent ici que les théorie physiques à partir desquelles sont fondées les HM peuvent servir à déterminer si la fonction calculée par un dispositif est une fonction calculable ou non. Leur thèse semble en particulier correcte dans le cas des o-machines proposées par Stannett [Stannett, 2003] et Calude [Calude, 2005]. La théorie de la radioactivité et l'interprétation standard de la mécanique quantique permettent en effet d'inférer, à partir du postulat de Born et de la désintégration radioactive, que les processus utilisés par les o-machines de Stannett et Calude ne sont pas pseudo-aléatoires, et donc qu'elles hyper-calculent.

Stannett a néanmoins soulevé un problème de taille à propos d'une vérification fondée sur la théorie de la radioactivité [Stannett, 2001a]. Il a en effet montré que la vérification d'une OMA utilisant cette théorie dans le but de produire une SNA est soumise à l'alternative suivante : soit l'hypothèse du caractère aléatoire de la désintégration radioactive est correcte et l'on peut af-

firmer que l'OMA est un véritable générateur de SNA utilisant un échantillion radioactif ; soit l'hypothèse est fausse et aucune expérience menée en conformité avec la théorie radioactive ne permet de réfuter la thèse selon laquelle l'OMA hyper-calcule. En d'autres termes, cette alternative signifie que toute expérience ayant pour objet de réfuter la puissance hyper-calculatoire de l'HM est du même coup une réfutation de la théorie radioactive. Ainsi, même si la théorie de la radioactivité explique pourquoi un dispositif fondée sur cette théorie devrait hyper-calculer, il est impossible d'en obtenir une preuve expérimentale. Il semble donc que cette théorie ne puisse pas apporter de réponse robuste au problème de vérifier les résultats fournis par une OMA. Toufefois, le résultat de Stannett ne remet pas en cause toute tentative de vérification à partir des théories physiques puisqu'il ne s'applique que dans le cas des *o*-machines fondées sur la théorie de la radioactivité. Des recherches complémentaires devront par exemple être effectuées au niveau de la théorie quantique si l'on souhaite généraliser le résultat de Stannett à l'OMA de Calude. Les résultats d'Abbott, Calude, Conder et Svozil inclinent néanmoins à penser que les OM ne sont pas soumises au problème soulevé par Stannett [Abbott et al., 2012].

Vérification à partir de tests empiriques. La dernière technique heuristique ne concerne que les OMA. Elle consiste à effectuer des tests empiriques sur les SNA produites par les oracles afin d'augmenter la probabilité que ces suites sont aléatoires et donc non Turing-calculables. Il existe par exemple certaines propriétés propres aux SNA - être normale, suivre la loi des grands nombres - qui pourraient être confirmées à l'aide de tests empiriques sur les suites produites par les oracles aléatoires.

La vérification du caractère aléatoire d'une suite à l'aide de tests empiriques soulève cependant de nombreux problèmes qui réduisent considérablement les chances liées au succès de cette technique :

1. D'après Martin-Löf, démontrer qu'une suite est aléatoire est une entreprise impossible pour deux raisons. La première est qu'il existe une infinité de tests devant être effectués si l'on souhaite démontrer qu'une suite est aléatoire. La seconde est que même s'il existe en théorie un

le test universel nécessaire et suffisant pour démontrer le caractère aléatoire d'une suite, ce test nous est en pratique inaccessible [Martin-Löf, 1966].

2. Les générateurs de nombres pseudo-aléatoires qui sont actuellement créés produisent des suites de nombres qui sont en pratique impossibles à distinguer de suites aléatoires fournies par des générateurs tels que Quantis [Cleland, 2004]. Cela signifie que l'on ne parvient déjà plus à faire la différence entre suites pseudo-aléatoires produites par des algorithmes et suites aléatoires produites par des processus physiques qui sont considérés comme aléatoires.

Malgré ces difficultés, il n'est toutefois pas impossible que l'on dispose un jour de fondements raisonnables permettant d'affirmer avec une forte probabilité que certaines suites de nombres sont aléatoires.

4 Conclusion

Le problème de ce chapitre fut de déterminer s'il est possible de démontrer la TPHC. Démontrer la TPHC nécessite en particulier de pouvoir vérifier qu'un dispositif - que l'on suppose être une HM - calcule au moins une fonction non Turing-calculable. Le problème a par conséquent été énoncé de la manière suivante : supposons que l'on dispose d'un dispositif, peut-on vérifier qu'il hyper-calcule ?

Ma réponse à cette question fut négative car il est selon moi impossible de vérifier que les valeurs fournies par un dispositif - ou une machine physiquement construite - correspondent aux valeurs de fonctions non Turing-calculables. En ce sens, la TPHC ne peut pas être démontrée puisqu'il est impossible de vérifier que l'on a construit une HM. Plus précisément, j'ai construit mon argumentation en trois étapes.

J'ai tout d'abord commencé par étudier de quelle façon on peut affirmer qu'un système physique calcule. J'ai ainsi présenté l'analyse standard du calcul qui fournit certains critères pour attribuer le calcul aux systèmes physiques. La principale particularité de cette analyse est qu'elle conduit

au pancomputationnalisme, thèse selon laquelle tous les systèmes physiques calculent. Même si l'analyse standard ne remet pas en cause l'hyper-calcul au sein des systèmes physiques, j'ai néanmoins tenté de montrer qu'elle ne pouvait pas être utilisée dans le but d'affirmer qu'un système physique hyper-calcule. Pour ce faire, j'ai d'abord expliqué que sous certaines conditions cette analyse conduit à un hypercomputationnalisme, à savoir à l'affirmation que tous les systèmes physiques hyper-calculent. J'ai ensuite montré que l'hypercomputationnalisme ne pouvait être soutenue et donc qu'une approche différente de l'analyse standard était nécessaire pour résoudre le problème de la vérification.

Cette approche consiste (1) à étudier les méthodes et techniques que les scientifiques utilisent pour vérifier qu'un ordinateur calcule une fonction donnée ; et (2) à appliquer ces méthodes aux machines afin de déterminer si elles hyper-calculent. Suite à l'analyse de cette approche, je suis parvenu à une première conclusion : il est impossible de vérifier parfaitement les résultats fournis par une machine car seule une vérification partielle fondée sur les probabilités est possible. Cette conlusion a eu pour conséquence de redéfinir le problème de la vérification en requérant non plus une vérification totale mais une vérification partielle des résultats fournis par une machine. En particulier, j'ai montré que le problème de la vérification partielle était résoluble pour le calcul effectif car certaines techniques informatiques, mathématiques et physiques pouvaient être utilisées dans le but d'affirmer qu'un ordinateur calcule une fonction donnée. J'ai en outre défendu que la possibilité de vérifier partiellement les résultats fournis par les ordinateurs émanait d'une de leurs caractéristiques, à savoir leur utilisabilité. A partir des travaux de Piccinini, j'ai défini l'utilisabilité d'une machine comme un ensemble de contraintes devant être satisfait par une machine afin de pouvoir affirmer que ses résultats sont partiellement vérifiables.

La dernière étape de mon argumentation a consisté à évaluer si l'hyper-calcul était utilisable, c'est-à-dire si les HM proposées par les scientifiques pouvaient satisfaire les contraintes inscrites au sein de la condition d'utilisabilité. J'ai ici défendu qu'aucune de ces HM ne satisfaisaient l'ensemble des contraintes d'utilisabilité. Je me suis en particulier focalisé sur les OMA

qui sont selon moi les propositions les plus prometteuses pour démontrer la TPHC. A partir de l'impossibilité de vérifier que les *o*-machines hyper-calculent, j'ai conclu qu'il était fort probable que la TPHC soit indémontrable. Plus précisément, puisqu'il n'existe aucun moyen permettant de vérifier que les HM récemment proposées hyper-calculent, aucune preuve de la TPHC ne peut être apportée. Autrement dit, la TPHC est selon moi condamnée à revêtir le statut de thèse et non d'énoncé scientifique.

Conclusion générale

Ce travail a eu pour but de défendre la thèse suivante : les ordinateurs - dispositifs ayant pour fonction d'exécuter des calculs - sont indispensables pour établir les limites de ce qui est calculable.

En soutenant cette thèse, je me suis ainsi opposé à une position soutenue depuis la seconde moitié du XX^{e} siècle selon laquelle les limites du calcul sont en principe indépendantes des limites des ordinateurs. Plus précisément, cette position affirme que les ordinateurs sont en principe négligeables lorsque l'on cherche à définir les limites de ce qui est calculable, et cela même s'ils sont en pratique indispensables pour calculer ou mémoriser des données. Les ordinateurs sont négligeables car leurs capacités calculatoires - vitesse de calcul, espace de stockage, *etc.* - n'ont aucune influence sur les limites en principe définies par les logiciens des années 1930.

Plus généralement, définir des limites en principe indépendantes des ordinateurs a pour conséquence de ne plus faire référence aux types de calculateurs - humains ou machines - traditionnellement différenciés par leurs capacités. Il est ainsi possible de définir les limites du calcul sans les reconduire à des distinctions liées aux différents calculateurs - dans le cas présent, une distinction entre ce qui est calculable par l'être humain et ce qui est calculable par l'ordinateur. L'étude du calculateur - qu'il soit humain ou machine - n'est donc plus pertinente pour étudier les limites du calcul.

Dans ce travail, je me suis cependant opposé à la thèse selon laquelle la construction d'ordinateurs n'a aucune influence sur les limites en principe du calcul. J'ai voulu montrer qu'il est indispensable d'étudier les ordinateurs pour établir les limites du calcul, que ces limites soient définies en principe ou en pratique. En ce sens, j'ai tenté de remettre en cause la distinction en

principe / en pratique sur laquelle se fondent les logiciens des années 1930 pour établir des limites indépendantes des avancées technologiques.

Pour replacer le concept d'ordinateur au cœur de l'étude des limites du calcul, ma stratégie a été de fournir des arguments en faveur de la thèse selon laquelle certains types d'ordinateurs - appelés *hyper-machines* (HM) - pourraient être capables d'*hyper-calculer*, à savoir de franchir la *barrière de Turing* - limite entre ce qui est en principe calculable et ce qui ne l'est pas. La vérité de cette thèse modifierait profondemment les limites du calcul érigées dans les années 1930 car elle rendrait de fait la barrière de Turing dépendante des avancées technologiques et en particulier des ordinateurs.

Dans cette conclusion générale, je reviens dans un premier temps sur les principaux arguments qui m'ont permis de défendre cette thèse. Je présente dans un second temps les défis qui devront être relevés dans le futur pour la démontrer.

Thèses défendues

Au total, quatre thèses ont été défendues dans ce travail, chacune faisant l'objet d'un chapitre. Les trois premières sont en faveur de la thèse physique de l'hyper-calcul (TPHC) selon laquelle il est possible de construire physiquement une HM. La quatrième soulève en revanche certaines difficultés liées à une possible démontration de la TPHC. Je retrace dans ce qui suit le chemin argumentatif des trois premières thèses ; la dernière sera quant à elle exposée dans la seconde partie de cette conclusion.

L'hyper-calcul est logiquement possible

Pour défendre la thèse selon laquelle la construction physique d'une HM pourrait remettre en cause la barrière de Turing, j'ai tenté de montrer que l'hyper-calcul est une notion logique consistante - *cf.* chapitre 1. La construction physique d'une HM est en effet dépendante de la possibilité logique de l'hyper-calcul puisque la possibilité logique est une condition nécessaire à la possibilité physique. Autrement dit, si l'hyper-calcul est logiquement possible

alors la question de sa possibilité physique peut être soulevée ; mais si l'hyper-calcul se révèle logiquement contradictoire, toute tentative de construction physique d'une HM est vouée à l'échec.

Plus précisément, je me suis élevé contre deux objections ayant pour cible la possibilité logique de l'hyper-calcul :

1. Une première objection qui est énoncée contre la notion même d'hyper-calcul. Cette objection affirme que l'hyper-calcul est logiquement contradictoire car elle est en opposition avec la thèse de Church-Turing (TCT).
2. Une seconde objection qui est quant à elle invoquée contre les processus infinis utilisés par les HM pour calculer des fonctions non Turing-calculables.

Ces deux objections sont selon moi fallacieuses. Tout d'abord, la première objection ne tient pas car l'hyper-calcul n'est pas en opposition avec la TCT. En effet, l'hyper-calcul et le calcul effectif doivent être considérés comme des définitions alternatives de la notion de calcul. En ce sens, il n'existe pas *une* définition de la notion de calcul mais plusieurs définitions à partir desquelles on peut formaliser des machines théoriques capables de calculer différents types de fonctions. Tandis que les MT et les URM (*Unlimited Register Machine*) calculent uniquement des fonctions Turing-calculables, les machines de Turing accélérantes, infinies et à oracles peuvent quant à elles calculer des fonctions non Turing-calculables. Il n'est donc pas contradictoire que le calcul effectif et l'hyper-calcul soient logiquement possibles.

La seconde objection, qui comprend deux arguments contre les processus infinis exécutés par les HM, ne parvient pas non plus à prouver que l'hyper-calcul est contradictoire. Dans un premier temps, les paradoxes de l'infini - sur lesquels s'appuie le premier des deux arguments - ne permettent pas de conclure qu'il est contradictoire d'exécuter une infinité d'étapes en un temps fini, processus appelé *supertask* (ST). Dans un second temps, certaines solutions peuvent être proposées afin d'échapper au second argument affirmant que les machines de Turing accélérantes (MTA) - HM pouvant exécuter un ST - sont logiquement contradictoires en vertu du fait qu'elles sont capables de calculer leurs propres fonctions arrêt et diagonale.

L'être humain ne peut pas franchir la barrière de Turing sans construire physiquement une HM

Supposons que l'être humain soit capable d'hyper-calculer de lui-même sans l'aide d'aucun dispositif, est-ce que la construction physique d'une HM replacerait les ordinateurs au centre de l'étude des limites du calcul ? Non, car le calculateur humain n'aurait dans ce cas nul besoin des ordinateurs pour franchir la barrière de Turing ; ces derniers seraient donc de nouveau mis à l'écart. Dans le but de montrer que les limites du calcul ne sont pas indépendantes de la construction des ordinateurs, j'ai par conséquent défendu la thèse selon laquelle la construction d'une HM est nécessaire pour hyper-calculer ; autrement dit que l'être humain ne peut pas hyper-calculer sans l'aide d'une HM physiquement construite - *cf.* chapitre 2.

La position selon laquelle les limites du calcul ne sont pas indépendantes de la construction des ordinateurs va cependant à l'encontre de ce qui est soutenu depuis la seconde moitié du XX^{e} siècle. L'analyse des logiciens des années 1930 a en effet permis de montrer que même si les ordinateurs sont en pratique indispensables, ils n'ont en principe aucune influence sur les limites du calcul. Pour comprendre ce point fondamental, j'ai commencé par définir précisément ce qu'est un ordinateur car la nécessité d'utiliser un ordinateur pour calculer dépend de ce que l'on considère comme étant un ordinateur. J'ai ensuite expliquer (1) que les ordinateurs sont en pratique nécessaires pour calculer ; et (2) qu'ils sont en principe contingents lorsque l'on considère les calculs pouvant être exécutés par un être humain.

Plus précisément, la puissance de calcul d'un ordinateur le rend en pratique indispensable pour effectuer des calculs qui demandent trop de ressources physiques. Toutefois, cette puissance n'est en principe plus nécessaire. D'une part, les ordinateurs sont des dispositifs universels au sens où les calculs qu'ils peuvent effectuer sont exactement ceux pouvant être exécutés par une MT. D'autre part, les travaux de Turing ont permis de montrer que les calculs pouvant être effectués par une MT peuvent être exécutés en principe par un être humain. L'ordinateur n'est donc pas nécessaire en principe puisqu'un être humain peut exécuter tous les calculs d'un ordinateur.

Contrairement à la construction des ordinateurs, j'ai ensuite défendu la thèse selon laquelle la construction d'une HM est en pratique et en principe indispensable pour hyper-calculer. Mon principal argument a été d'affirmer que l'être humain n'est en principe pas capable d'hyper-calculer, et donc qu'il ne peut pas faire abstraction des ordinateurs comme dans le cas du calcul effectif. Premièrement, j'ai expliqué que mon argument reposait sur la vérité de la *thèse des hyper-cerveaux*, thèse selon laquelle le cerveau humain peut hyper-calculer. Deuxièmement, j'ai tenté de défaire cette thèse en montrant que les principaux arguments en sa faveur n'étaient pas capables de la démontrer.

Les HM fondées sur l'utilisation de l'aléatoire comme source d'hyper-calcul sont prometteuses

La thèse selon laquelle les HM sont indispensables pour établir les limites de ce qui est calculable ne peut être validée *que* si une HM est physiquement construite. Il se pourrait en effet que la construction physique d'une HM soit physiquement impossible. Si tel était le cas, la barrière de Turing resterait à la fois intacte et indépendante de la construction des ordinateurs. L'hyper-calcul repose ainsi *in fine* sur la construction physique d'une HM.

Toutefois, la position selon laquelle il est possible de construire une HM - appelée *thèse physique de l'hyper-calcul* (TPHC) - ne fait pas l'unanimité dans la communauté scientifique. J'ai tout de même soutenu que certaines propositions introduites en vue de démontrer la TPHC ont des chances d'aboutir - *cf.* chapitre 3. Plus précisément, j'ai essayé de montrer que les propositions de Stannett [Stannett, 2003] et de Calude [Calude, 2005] fondées sur la notion d'aléatoire sont plus prometteuses que celles de Davies [Davies, 2001], Shagrir & Pitowsky [Shagrir and Pitowsky, 2003] et Kieu [Kieu, 2001] issues des théories newtonienne, relativiste et quantique.

J'ai commencé par expliquer que j'étais en accord avec la conclusion de la communauté scientifique selon laquelle il fort peu probable que les HM de Davies, de Shagrir & Pitowsky et de Kieu soient un jour construites. Pour chacune de ces propositions, voici le principal obstacle à leur possible

construction :

HM newtonienne de Davies Le problème majeur est que la propriété physique essentielle sur laquelle se fonde cette HM, à savoir qu'il n'existe pas de limite physique à la division de l'espace et du temps, n'est pas une propriété que possède le monde réel.

HM relativiste de Shagrir & Pitowsky Même si la théorie relativiste n'est pas infirmée, il n'existe actuellement aucune preuve de l'existence - au sens physique - des espace-temps de Malament-Hogarth censés permettre à l'HM d'hyper-calculer.

HM quantique de Kieu L'HM soulève de nombreux problèmes physiques liés aux difficultés (1) d'implémenter physiquement certains opérateurs hamiltoniens ayant un nombre infini de niveaux d'énergie ; et (2) d'obtenir physiquement les états fondamentaux.

J'ai ensuite défendu que les HM utilisant l'aléatoire comme source d'hypercalcul sont plus prometteuses que leurs analogues newtoniennes, relativistes et quantiques. La défense de ces HM s'est faite à partir de trois arguments :

1. **Les critiques le plus souvent énoncées à l'encontre de la notion de suite de nombres aléatoires (SNA) peuvent être surmontées**. Ces critiques consistent (1) à reprocher à la notion de SNA de n'être pas correctement définie d'un point de vue mathématique ; et (2) à affirmer qu'il est impossible de produire des SNA.

2. **Le dispositif nécessaire pour faire fonctionner l'HM est construit.** Ce dispositif est en effet essentiellement composé de deux parties pouvant être construites : un ordinateur et un générateur de nombres aléatoires tel que *Quantis* de l'entreprise *ID Quantique*.

3. **Les hypothèses sur lesquelles l'HM est fondée sont confirmées par l'expérience**. Cela signifie que les expériences physiques qui sont effectuées en continu par les scientifiques sont en accord avec les hypothèses statistiques sur lesquelles reposent l'HM.

Problèmes et recherches futures

Si les thèses précédentes sont en faveur de la TPHC, la démonstration de cette thèse soulève toutefois certaines difficultés. J'ai en particulier expliqué dans le quatrième et dernier chapitre de mon travail que la vérification des valeurs fournies par une HM que l'on aurait physiquement construite pourrait être un obstacle à la démonstration de la TPHC. Plus précisément, il se pourrait que l'on ne puisse pas vérifier qu'une HM a bien été construite, résultat qui, s'il s'avérait exact, condamnerait la TPHC à revêtir le statut de thèse et non d'énoncé scientifique. Le problème de la vérification semblerait ainsi donner raison à Davis : l'hyper-calcul est un mythe au même titre que le mouvement perpétuel avant lui [Davis, 2004]. Mais s'il est vrai que la vérification des HM est un véritable problème pour l'hyper-calcul, les recherches menées actuellement en informatique, en physique et en ingénieurie portent à croire que le calcul de fonctions non Turing-calculables pourrait être un jour une réalité.

Le problème de la vérification : un obstacle à la démonstration de la TPHC

Démontrer la TPHC nécessite de pouvoir vérifier qu'un dispositif - que l'on suppose être une HM - calcule au moins une fonction non Turing-calculable. Le problème de la vérification a par conséquent été énoncé de la manière suivante : supposons que l'on dispose d'un dispositif, peut-on vérifier qu'il hyper-calcule ?

Pour répondre à cette question, j'ai tout d'abord commencé par étudier de quelle façon on peut affirmer qu'un système physique calcule. J'ai ainsi présenté l'analyse standard du calcul qui fournit certains critères pour attribuer le calcul aux systèmes physiques. Si l'analyse standard ne remet pas en cause l'hyper-calcul au sein des systèmes physiques, j'ai néanmoins tenté de montrer qu'elle ne pouvait pas être utilisée dans le but d'affirmer qu'un système physique hyper-calcule.

J'ai ensuite évalué une seconde approche qui consiste (1) à étudier les

méthodes et techniques que les scientifiques utilisent pour vérifier qu'un ordinateur calcule une fonction donnée ; et (2) à appliquer ces méthodes aux machines afin de déterminer si elles hyper-calculent. Suite à cette évaluation, je suis parvenu à une première conclusion : il est impossible de vérifier parfaitement les résultats fournis par une machine car seule une vérification partielle est possible. Cette conlusion a eu pour conséquence de redéfinir le problème de la vérification en requérant non plus une vérification totale mais une vérification partielle des résultats fournis par une machine. En particulier, j'ai montré que le problème de la vérification partielle était résoluble pour le calcul effectif car certaines techniques pouvaient être utilisées dans le but d'affirmer qu'un ordinateur calcule une fonction donnée. J'ai en outre défendu à partir des travaux de Piccinini que la possibilité de vérifier partiellement les résultats fournis par les ordinateurs émanait de leur utilisabilité.

La dernière étape de mon argumentation a consisté à évaluer si les HM étaient utilisables, condition nécessaire afin de pouvoir vérifier partiellement leurs résultats. Malheureusement les HM actuellement proposées ne sont d'après moi pas utilisables ; il est donc impossible de vérifier - même partiellement - qu'elles hyper-calculent. Puisqu'il n'existe aucun moyen permettant de vérifier que les HM récemment proposées hyper-calculent, aucune preuve de la TPHC ne peut donc être apportée. Autrement dit, la TPHC est selon moi condamnée à revêtir le statut de thèse et non d'énoncé scientifique.

Briser le mythe de l'hyper-calcul

L'impossibilité de vérifier qu'une HM hyper-calcule étant un sérieux obstacle à la démonstration de la TPHC, est-ce que le calcul de fonctions non Turing-calculables est encore envisageable ? Autrement dit, est-ce que Davis a vu juste en affirmant que l'hyper-calcul est un mythe au même titre que le mouvement perpétuel ?

Donner raison à Davis serait seloi moi prématuré car même si personne ne peut garantir qu'il est possible de franchir la barrière de Turing, personne ne peut non plus prétendre que c'est impossible. Plus précisément, s'il est démontré que le mouvement perpétuel est impossible du fait de son incom-

patibilité avec les lois physiques - en particulier avec celles énoncées par la thermodynamique - aucune preuve de l'incompatibilité de l'hyper-calcul avec les lois régissant le monde réel n'a été apportée. Au contraire, je pense que les recherches menées actuellement laissent entrevoir plusieurs pistes prometteuses qui pourraient bien briser le mythe de l'hyper-calcul.

Premièrement, un projet de recherche sur le problème de la vérification dirigé par Stannett va être réalisé à l'Université de Sheffield. Ce projet consiste à développer des techniques pour identifier, vérifier et tester les comportements supposés hyper-calculatoires. Il consiste plus précisément à décrire et à vérifier les systèmes qui exploitent la puissance d'oracles physiques et logiques situés à divers niveaux de la hiérarchie arithmétique. Même s'il n'existe pas de technique générale pour démontrer qu'un système hyper-calcule, le but de ce projet est de développer une méthode simple mais complète capable de spécifier de tels systèmes. L'idée de base est d'utiliser une version spécifique des X-machines, HM capables de résoudre des problèmes indécidables énoncés sous la forme de fonctions continues [Eilenberg, 1974]. Ce type de machines est particulièrement adapté à ce projet pour trois principales raisons :

1. Elles permettent de distinguer les aspects de contrôle et de traitement d'un système donné afin de les étudier séparément [Stannett, 2001a].
2. Elles permettent de décrire des systèmes à partir d'un nombre fini d'états computationnels tout en préservant les comportements hyper-calculatoire de la machine [Stannett and Németi, 2012].
3. Elles permettent de modéliser de nombreux systèmes différents [Stannett, 2013].

Deuxièmement, il existe d'autres propositions de construction d'HM différentes de celles que j'ai analysées tout au long de ce travail. Ces propositions font elles aussi l'objet de recherches, bien que ces dernières soient moins avancées que celles menées sur les HM relativistes et quantiques qui ont été introduites au chapitre 3. Voici des exemples d'HM actuellement étudiées :

– Les *stream X-machines* [Laycock, 1993]. Actuellement utilisées dans le domaine du développement cellulaire [Bell and Holcombe, 1996], elles pourraient utiliser les propriétés des X-machines pour hyper-calculer.

- Les *signal machines* [Durand-Lose, 2011]. La possibilité d'hyper-calculer repose sur la cardinalité des signaux produits par ces machines : si ces signaux sont de cardinalité finie, elles calculent uniquement les fonctions Turing-calculables ; si cette cardinalité est infinie, elles peuvent en revanche franchir la barrière de Turing.
- Les machines euclidiennes [Mycka et al., 2008]. Ces machines produisent et résolvent des problèmes indécidables à partir de constructions à l'aide d'une règle et d'un compas.
- Les HM fondées sur l'étude d'agents biologiques tels que les membranes cellulaires [Calude and Păun, 2004]. L'étude des membranes cellulaires pourrait en particulier mener à l'affirmation selon laquelle le cerveau humain est capable d'hyper-calculer.

Le projet dirigé par Stannett conjugué à la diversité des recherches menées en parallèles sur les HM citées plus haut atteste selon moi du fait que l'hyper-calcul ne possède pas le même statut que le mouvement perpétuel. Contrairement à ce dernier, les recherches actuelles sur l'hyper-calcul sont la preuve que la question liée à la possibilité de franchir la barrière de Turing n'est pas tranchée.

Que la réponse à cette question soit positive ou négative, elle aurait des conséquences philosophiques importantes à propos des limites du calcul. Une réponse positive rendrait ainsi indispensable l'utilisation des ordinateurs pour étudier les limites du calculable. Les ordinateurs ne seraient plus seulement nécessaires pour repousser les limites de la faisabilité pratique - qui est dépendante des ressources physiques utilisées lors des calculs - mais deviendraient aussi nécessaires pour repousser celles de la faisabilité en principe - qui fait abstraction des ressources physiques. Les ordinateurs donneraient accès à des calculs différents des calculs effectifs, c'est-à-dire à des calculs qui ne peuvent en principe pas être exécutés par un être humain ou par un ordinateur actuel.

Une réponse négative aurait quant à elle pour conséquence de renforcer la barrière de Turing. Dans ce cas, ni les ordinateurs actuels ni les ordinateurs de demain ne seraient une menace pour la thèse selon laquelle les fonctions calculables sont les fonctions effectivement calculables, à savoir les

fonctions calculables par MT. Les limites en principe du calcul resteraient par conséquent indépendantes des facteurs technologiques.

Annexe A

La machine de Turing

L'exposition qui suit est fondée sur des références standards de la théorie de la calculabilité telles que [Davis, 1958] et [Rogers, 1987].

1 Machine de Turing

Une () est une formalisation mathématique qui est constituée d'un ruban infini - dans les deux sens - d'une table d'instructions ou programme, et d'une tête de lecture / écriture. Le ruban est divisé en un nombre infini de cellules et la tête de lecture peut écrire un symbole à l'intérieur de chaque cellule. Les symboles sont des éléments d'un ensemble $A = \{S_0, S_1, \ldots, S_n\}$ $n \geq 1$, qui est appelé l'*alphabet*. Par convention S_0 est le symbole représentant le vide, c'est-à-dire que lorsque la tête de lecture écrit ce symbole dans une cellule, elle efface le symbole qui était inscrit dans cette même cellule. A tout moment, la MT est dans l'état q_i qui est un élément d'un ensemble fini $Q = \{q_0, q_1, \ldots, q_r\}$ $r \geq 1$. Le programme est quant à lui utilisé pour déterminer la prochaine exécution de la MT à un moment donné. Plus précisément, ce que va faire la MT dépend de son état actuel et du symbole qui est en train d'être lu par sa tête de lecture. Si aucune action n'a été spécifiée pour une combinaison d'état et de symbole, la MT s'arrête. Généralement, le programme de la MT est défini par une suite finie de quadruples qui sont des cas particuliers d'expressions.

Définition (Expression)
Une expression est une suite de symboles choisie à partir des éléments de la liste $q_0, q_1, \ldots, S_0, S_1, \ldots, D, G$.
Un quadruple peut prendre l'une des formes suivantes :

$$q_i S_j S_k q_l \tag{A.1}$$

$$q_i S_j L q_l \tag{A.2}$$

$$q_i S_j R q_l \tag{A.3}$$

Notons que $R, L \notin A$. Le quadruple (A.1) par exemple indique que si la MT est dans l'état q_i et que la cellule qu'elle est en train de lire contient le symbole S_j, alors la tête de lecture remplace S_j par S_k et entre dans l'état q_l. Les symboles R, L signifient respectivement que la tête de lecture se déplace vers la cellule à droite ou à vers la cellule à gauche de la cellule où elle se trouve.

La MT est utilisée pour calculer les valeurs de fonctions $f(x_1, \ldots, x_n)$ qui prennent leurs valeurs dans $\mathbb{N}^n$. Chaque argument $x_i \in \mathbb{N}$ est représenté sur le ruban en inscrivant le symbole S_1 à l'intérieur de $x_i + 1$ cellules. Les représentations des arguments sont séparées par une cellule vide - le symbole S_0 est donc inscrit dans chaque cellule vide. Toutes les autres cellules autres que celles utilisées pour la réprésentation des arguments sont vides. Par convention, les symboles 1 et 0 sont utilisés respectivement à la place de S_1 et de S_0. Par exemple, la représentation des trois arguments 3, 4, 2 sera

1111011111011

La MT commence son calcul en étant dans l'état q_0 et sa tête de lecture est placée sous le premier symbole 1 de la suite d'arguments. Si la MT a atteint une situation pour laquelle aucun ou plus d'un quadruple est applicable, alors la MT s'arrête. Une fois l'arrêt de la MT, le résultat du calcul est égal au nombre de cellules pour lesquelles le symbole 1 est inscrit.

Définition (Fonction calculée par une MT)
Soit M *une MT et soit* $\Psi_M^n(x_1, \ldots, x_n)$ *une fonction partielle à* n *arguments.*

On dit que M calcule Ψ_M^n si pour chaque tuple $(m_1, \ldots, m_n)$ d'arguments, M s'arrête après un nombre fini d'étapes. Si M ne s'arrête pas sur un tuple $(k_1, \ldots, k_n)$, alors Ψ_M^n est non définie pour ce tuple. On dit que M calcule la fonction f ou que f est Turing-calculable si pour tout $(x_1, \ldots, x_n)$, $\Psi_M^n(x_1, \ldots, x_n)$ est définie et égale à $f(x_1, \ldots, x_n)$.

La fonction Ψ_M^n est la fonction calculée par M à partir de son programme. De ce point de vue, des MT différentes calculent des fonctions différentes ; il est donc pour l'instant impossible de soutenir que la MT est une formalisation capable de calculer toutes les fonctions intuitivement calculables - il faudrait pour cela autant de MT que de fonctions calculables, à savoir une infinité. Ce qu'il manque c'est une définition de la MT qui englobe le calcul de chaque MT particulière. Un des grands résultats de la calculabilité est justement d'avoir proposé une telle définiton.

2 Machine de Turing universelle

Il est effet possible de concevoir une MT appelée () qui prend en entrée à la fois la description d'un programme d'une MT particulière et ses arguments. Pour ce faire, il est nécessaire de coder par un entier naturel la donnée en entrée de la MTU composée (1) du programme de la MT ; et (2) de ses arguments. Voici la marche à suivre. On commence par associer à chaque symbole de base de la MT un nombre impair supérieur ou égal à 3 :

3	5	7	9	11	13	15	17	19	21	...
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	
R	L	S_0	q_0	S_1	q_1	S_2	q_2	S_3	q_3	...

Pour chaque i, S_i est codé par $4i + 7$ et q_i par $4i + 9$. Dans le but de définir le code d'une MT, il est tout d'abord nécessaire de définir les codes des expressions et des ensembles d'expressions.

Définition (Nombre de Gödel)

Supposons que M est une chaîne de symboles $\gamma_1, \ldots, \gamma_n$ et que $a_1, \ldots, a_n$ sont les entiers associés à chacun de ces symboles. Le nombre de Gödel de M est

l'entier

$$Gn(M) = \prod_{k=1}^{n} (Pr(k))^{a_k}$$

où $Pr(k)$ est le k-ième nombre premier $\neq 1$.

Par exemple, si $M = q_1 S_0 S_2 q_1$ alors $Gn(M) = 2^{13}.3^7.5^{15}.7^{13}$. La méthode présentée ci-desssus est souvent appelée *codage à la Gödel* et peut s'appliquer à tous les éléments des ensembles dénombrables. Néanmoins, le véritable codage que Gödel a utilisé dans son mémoire est plus complexe car il permet de coder une chaîne et de retrouver la chaîne codée grâce à une fonction - la fonction β - qui est calculable par MT [Gödel, 1931]. Ce qui est important est toutefois de comprendre qu'il existe une méthode de calcul permettant de coder ainsi que de retrouver le code d'une suite de symboles.

Définition (*Gn* d'un ensemble fini d'expressions)
Supposons que $M_1, \ldots, M_n$ est une suite finie d'expressions. Le nombre de Gödel de cette suite est l'entier

$$\prod_{k=1}^{n} (Pr(k))^{Gn(M_k)}$$

Définition (*Gn* d'une MT)
Supposons que $M_1, \ldots, M_n$ est n'importe quel arrangement sans répétition de quadruples de MT M. Le nombre de Gödel de la suite $M_1, \ldots, M_n$ est le Gn de M.

Clairement, une MT qui est constituée de n quadruples possède $n!$ Gn. Il est à présent possible de présenter une définiton de la MTU.

Définition (Machine de Turing universelle)
Une MTU U est une MT qui peut être utilisée pour calculer toute fonction à un argument calculée par une MT ordinaire. Plus précisément, cela signifie que si la MT M a pour Gn m et calcule la fonction f, alors

$$\Psi_U^2(m, x) = f(x) = \Psi_M^1(x)$$

Autrement dit, si le nombre m est écrit sur le ruban de U et qu'il est suivi du nombre x, U calculera le nombre $\Psi^1_M(x)$.

En généralisant, la MTU peut être utilisée pour calculer les fonctions à n arguments calculées par MT [Davis, 1958, p. 65]. Même si je ne rentre pas plus dans les détails du fonctionnement de la MTU, cela ne pose pas de problème pour la suite du travail. L'intérêt fondamental de la MTU réside dans la possibilité de parler des fonctions calculables par MTU - ou par convention par MT - sans devoir exhiber une MT particulière pour chaque fonction.

3 Fonctions calculables par machine de Turing

Précisons maintenant quelles fonctions peuvent être calculées par une MT. Pour ce faire, Kleene a défini une hiérarchie nommée qui permet de classer toutes les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ [Kleene, 1943]. A partir de ce classement, il est possible de montrer quelles fonctions sont Turing-calculables.

Définition (Hiérarchie arithmétique)

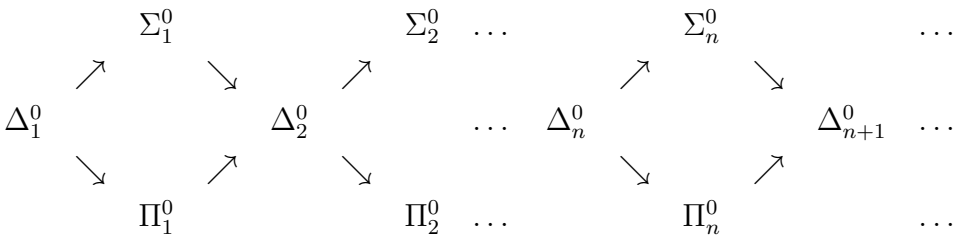
1. *Soit* Σ^0_0 *la classe de tous les sous-ensembles récursifs de* $\mathbb{N}$. *Pour chaque* $n \in \mathbb{N}$, Σ^0_{n+1} *est la classe des ensembles récursivement énumérables pour un certain ensemble* $A \in \Sigma^0_n$. *Il suit que* Σ^0_1 *est la classe des ensembles récursivement énumérables.*
2. *Soit* Π^0_0 *la classe de tous les sous-ensembles de* $\mathbb{N}$ *dont les compléments sont dans* Σ^0_0. *Autrement dit,* $D \in \Sigma^0_0$ *si et seulement si son complémentaire dans* $\mathbb{N}$ *est dans* Σ^0_0. *La classe* Π^0_1 *est connue dans la littérature comme la classe des ensembles co-récursivement énumérables.*
3. *Soit* Δ^0_n *l'intersection des classes* Σ^0_n *et* Π^0_n, *c'est-à-dire,* $\Delta^0_n = \Sigma^0_n \cap \Pi^0_n$.

Les classes Σ^0_n, Π^0_n, Δ^0_n forment la hiérarchie arithmétique et possèdent les propriétés suivantes :

Proposition

1. $\Delta_n^0 \subset \Sigma_n^0$ *et* $\Delta_n^0 \subset \Pi_n^0$
2. $\Sigma_n^0 \subset \Sigma_{n+1}^0$ *et* $\Pi_n^0 \subset \Pi_{n+1}^0$
3. $\Sigma_n^0 \cup \Pi_n^0 \subset \Delta_{n+1}^0, \forall n \geq 1$

Pour résumer, les relations entre ces classes peuvent être représentées par le schéma ci-dessous :



Il est maintenant possible de définir d'après la hiérarchie arithmétique la classe des fonctions Turing-calculables.

Théorème

La classe des fonctions Turing-calculables est la classe Δ_1^0*, c'est-à-dire l'intersection entre la classe des fontions récursivement énumérables et celle des fonctions co-récursivement énumérables.*

Annexe B

Les fonctions récursives

Cette présentation est fondée sur [Kleene, 1936].

1 Définition

Définition (Classe des fonctions primitives récursives)
La classe des fonctions primitives récursives (fonctions PR) est le plus petit sous-ensemble de $\{f : \mathbb{N}^n \to \mathbb{N}\}_{n\in\mathbb{N}}$ *tel que :*

1. $Z : \mathbb{N}^{m+1} \to \mathbb{N}$ *définie par* $Z(n, x_1, \ldots, x_m) = 0$ *appartient à PR - fonction zéro.*
2. $S : \mathbb{N}^{m+1} \to \mathbb{N}$ *définie par* $S(n, x_1, \ldots, x_m) = n + 1$ *appartient à PR - fonction successeur.*
3. $Pr_i^m : \mathbb{N}^m \to \mathbb{N}$ *définie par* $Pr_i^m(x_1, \ldots, x_i, \ldots, x_m) = x_i$ *appartient à PR,* $1 \leq i \leq m$ *- fonction projection.*
4. *PR est close par composition, c'est-à-dire :*
   Si $h : \mathbb{N}^m \to \mathbb{N}$ *et, pour* $1 \leq i \leq m$, $g_i : \mathbb{N}^p \to \mathbb{N}$, *appartiennent à PR, alors* $f : \mathbb{N}^p \to \mathbb{N}$ *définie par* $f(x_1, \ldots, x_p) = h(g_1(x_1, \ldots, x_p), \ldots, g_m(x_1, \ldots, x_p))$ *appartient à PR.*
   On dit que f *est définie par composition à partir de* $h, g_1, \ldots, g_m$.
5. *PR est close par récurrence, c'est-à-dire :*
   Si $g : \mathbb{N}^m \to \mathbb{N}$ *et* $h : \mathbb{N}^{m+2} \to \mathbb{N}$ *appartiennent à PR, alors* $f : \mathbb{N}^{m+1} \to \mathbb{N}$ *appartient à PR, où f est définie par :*

$$f(0, x_1, \ldots, x_m) = g(x_1, \ldots, x_m)$$

$$f(n+1, x_1, \ldots, x_m) = h(f(n, x_1, \ldots, x_m), n, x_1, \ldots, x_m)$$

On dit que f est définie par récurrence à partir de g et h.

Définition ()
Une fontion $f : \mathbb{N}^m \to \mathbb{N}$ est primitive récursive ssi $f \in PR$, c'est-à-dire si $f = Z$ ou $f = S$ ou $f = Pr_i^m$ ou f s'obtient à partir des ces fonctions par un nombre fini d'applications de composition et / ou récurrence.

Définition (Minimisation)
Soit f une fonction totale à n arguments. L'opération de minimisation sur f notée $Mn[f]$ est définie par

$$Mn[f](x_1, \ldots, x_n) = \begin{cases} \text{le plus petit } y \text{ pour lequel } f(x_1, \ldots, x_n, y) = 0 \\ \text{non définie si un tel } y \text{ n'existe pas.} \end{cases}$$

Définition ()
Les fonctions $f : \mathbb{N}^m \to \mathbb{N}$ pouvant être obtenues à partir des fonctions PR et de l'opération de minimisation sont appelées fonctions récursives.

A partir de la définition des fonctions récursives, il est naturel d'étendre la notion de récursivité aux ensembles, relations et prédicats. Informellement, un ensemble est récursif si on dispose d'une procédure effective apour déterminer si un élément appartient ou non à l'ensemble :

Définition (Ensemble récursif)
Soit $A \subseteq \mathbb{N}$ un ensemble. On dit que A est primitif récursif - respectivement récursif - si sa fonction caractéristique χ_A est primitive récursive - respectivement récursive.

Définition (Fonction caractéristique)
La fonction caractéristique χ_A d'un ensemble A est définie par
$\chi_A(n_1, \ldots, n_p) = 1$ *si* $(n_1, \ldots, n_p) \in A$
$\chi_A(n_1, \ldots, n_p) = 0$ *sinon.*

La notion de fonction caractéristique permet aussi de définir les relations et les prédicats récursifs.

Définition (Relation récursive)
Une relation $R \subseteq \mathbb{N}^m$ est primitive récursive - respectivement récursive - si χ_R définie par

$\chi_R(x_1, \ldots, x_m) = 1$ *si* $(x_1, \ldots, x_m) \in R$

$\chi_R(x_1, \ldots, x_m) = 0$ *sinon.*

est primitive récursive - respectivementé récursive.

Définition (Prédicat récursif)
Si $P(x_1, \ldots, x_p)$ est une propriété portant sur les entiers $x_1, \ldots, x_p$ - on parle aussi de prédicat d'arité p - on dit que P est récursif primitive - respectivement récursif - si l'ensemble

$\{(x_1, \ldots, x_p)/(x_1, \ldots, x_p)$ *vérifie* $P\}$

est primitif récursif - respectivement récursif.

2 Exemple d'une fonction non récursive

Voici une preuve simple et élégante qui montre que la *fonction diagonale* n'est pas récursive.

Soient $\psi_1, \psi_2, \psi_3 \ldots$ une énumération des fonctions récursives - une telle énumération est possible [Davis, 1958]. Soient f la fonction définie par

$$f(x, y) = \begin{cases} 0 \text{ si } \psi_x(y) \text{ diverge pour } y \\ 1 \text{ autrement.} \end{cases}$$

et g la fonction diagonale définie par

$$g(x) = \begin{cases} 0 \text{ si } f(x, x) = 0 \\ \text{autrement diverge pour } x. \end{cases}$$

Montrons que g n'appartient pas à l'ensemble des fonctions récursives. Si g est calculable, $\exists i$ / $\psi_i(x) = g(x)$. Cela implique que $\psi_i(i) = 0 \Longleftrightarrow g(i) = 0 \Longleftrightarrow f(i, i) = 0 \Longleftrightarrow \psi_i(i)$ diverge pour i, ce qui est contradictoire. Ainsi, $g(x)$ n'est pas récursive.

Annexe C

Les machines RAM

Le but général de cette annexe est de présenter une des façons de montrer qu'un ordinateur actuel est universel, c'est-à-dire équivalent à une MT en termes de fonctions calculables. Cette équivalence est prouvée ci-dessous à partir du modèle formel des microprocesseurs appelé (*Random Access Machines*). Cette présentation est inspirée de [Shepherdson and Strurgis, 1963] et [Cooper, 2004].

1 Définition

Les machines RAM sont des machines théoriques qui modélisent les unités de calcul des ordinateurs actuels, à savoir les microprocesseurs. Ces machines sont constituées d'un programme formé d'une suite d'instructions et d'une suite finie de registres. Les seules données que peuvent manipuler les machines RAM sont des nombres entiers. Plus précisément, une machine RAM est composée

- d'un *programme* lui-même constitué d'une suite finie d'instructions numérotées à partir de 0 ;
- d'une suite infinie de *registres* numérotés à partir de 0 ;
- d'un registre spécial appelé *compteur* ;
- d'un registre spécial appelé *accumulateur* noté A ;
- d'un *ruban d'entrée* sur lequel la machine lit ses données ;

– d'un *ruban de sortie* sur lequel la machine écrit ses résultats.

Les registres

Les registres et l'accumulateur contiennent les données manipulées par la machine qui sont exprimées sous la forme d'entiers de tailles arbitraires. Le compteur contient quant à lui le numéro de la prochaine instruction à exécuter. De façon générale le registre n sera noté R_n. Enfin, lors du lancement du programme tous les registres sont initialisés à la valeur zéro.

Les instructions

Les instructions d'une machine RAM se divisent en quatre catégories :

1. Les manipulations de registres.
2. Les opérations arithmétiques.
3. Les ruptures de suites.
4. Les instructions d'entrée et de sortie.

Les manipulations de registres

Les deux instructions de manipulation de registres sont les instructions *chargement* et *mémorisation* que l'on note respectivement C et M. L'instruction C permet de charger une valeur dans l'accumulateur à partir d'un registre ou d'une valeur explicite donnée en paramètre. L'instruction M permet quant à elle de mémoriser une valeur provenant de l'accumulateur dans un registre. Voici maintenant les différentes syntaxes de ces deux instructions :

1. L'instruction C admet les trois syntaxes suivantes où $< n >$ désigne un entier :
   (a) $C \sharp < n >$ où la valeur de l'entier donné en paramètre est chargé dans l'accumulateur.
   (b) $C < n >$ où le contenu du registre R_n est chargé dans l'accumulateur.

(c) C $(< n >)$ où le contenu du registre R_p, où p est le contenu du registre R_n, est chargé dans l'accumulateur. Le registre R_n est dans ce cas utilisé comme un pointeur qui donne le numéro ou l'adresse du registre à charger.

2. L'instruction M admet les deux syntaxes suivantes où $< n >$ désigne un entier :

   (a) $M < n >$ où le contenu de l'accumulateur est mémorisé dans le registre R_n.

   (b) M $(< n >)$ où le contenu de l'accumulateur est mémorisé dans le registre R_p, où p est le contenu du registre R_n.

Les opérations arithmétiques

Les machines RAM ne possèdent que deux opérations arithmétiques qui ne prennent pas de paramètre. D'une part, l'opération $\nearrow$ augmente d'une unité le contenu de l'accumulateur. D'autre part, l'opération $\searrow$ diminue d'une unité le contenu de l'accumulateur si ce contenu est strictement positif ; autrement elle laisse le contenu de l'accumulateur inchangé.

Les ruptures de suites

Les machines RAM possèdent trois instructions de ruptures de suites, à savoir les instructions $\curvearrowright$, $\Rsh$ et Ⓢ :

1. L'instruction $\curvearrowright$ appelée *saut inconditionnel* prend en paramètre un entier qui désigne le numéro d'une instruction dans le programme. Après l'exécution de cette instruction, le fonctionnement du programme se poursuit à l'instruction dont le numéro est donné en paramètre.

2. L'instruction $\Rsh$ appelée *saut conditionnel* prend en paramètre un entier qui désigne le numéro d'une instruction dans le programme. Après l'exécution de cette instruction, le fonctionnement du programme se poursuit à l'instruction dont le numéro est donné en paramètre si le contenu de l'accumulateur est 0, autrement il se poursuit à l'instruction suivante.

3. L'instruction Ⓢ appelée *stop* ne prend pas de paramètre. Cette instruction provoque l'arrêt de la machine.

Les instructions d'entrée et de sortie

Les machines RAM ne possèdent que deux instructions d'entrée et de sortie nommées *lire* et *écrire* qui ne prennent pas de paramètre. L'instruction *lire* provoque la lecture d'un entier sur le ruban d'entrée et le résultat est placé dans l'accumulateur. Chaque lecture avance la tête de lecture de la machine afin de provoquer une lecture séquentielle des données du ruban lorsqu'une succession d'instructions *lire* est exécutée. L'instruction *écrire* provoque quant à elle l'écriture du contenu de l'accumulateur sur le ruban de sortie.

Fonctionnement

Le déroulement du programme de la machine RAM consiste à exécuter successivement les instructions du programme en commençant à l'instruction numéro 0. Après l'exécution d'une instruction différente d'une instruction de rupture de suites, l'instruction suivante est exécutée. Une instruction de saut indique directement quelle est l'instruction suivante à exécuter. Ce processus se poursuit jusqu'à l'exécution d'une instruction stop ou lorsqu'il n'y a plus d'instruction à exécuter.

Exemple de programme

Le programme suivant lit deux entiers x et y, calcule leur somme et l'écrit sur le ruban de sortie. Les commentaires sont écrits entre crochets.

0 *lire* [Lecture du premier entier x]
1 $M0$ [Mémorisation de x dans R_0]
2 *lire* [Lecture du second entier y]
3 $M1$ [Mémorisation de y dans R_1]
4 $C0$ [Début de la boucle de calcul]

5 ↱12 [Test de sortie de boucle]
6 ↘ [A chaque itération de la boucle, R_0 est diminué et R_1 est augmenté]
7 $M0$ [La boucle s'arrête quand R_0 contient 0]
8 $C1$ [R_1 contient alors la somme de x et y]
9 ↗
10 $M1$
11 ↷ 4 [Fin de la boucle de calcul]
12 $C1$
13 *écrire* [Affichage de la somme]
14 Ⓢ

2 Universalité

Je vais maintenant montrer que les machines RAM sont équivalentes aux MT. Cela signifie d'une part que toute fonction calculable par une machine RAM est calculable par une MT et d'autre part que toute fonction calculable par une MT est calculable par une machine RAM. La stratégie adoptée est de montrer (1) que le fonctionnement d'une MT peut être simulé par une machine RAM ; et (2) que le fonctionnement d'une machine RAM peut être simulé par une MT.

Simulation d'une MT par une machine RAM

L'idée de la simulation est la suivante. Le contenu du ruban de la MT va être mémorisé dans les registres de la machine RAM à partir de R_1. De cette manière, chaque symbole du ruban sera mis dans un registre. Le registre R_0 contiendra quant à lui la position de la tête de lecture de la MT, c'est-à-dire le numéro du registre qui contient le symbole lu par la tête de lecture. On associera tout au long de la procédure un code sous la forme d'un entier à chacun des symboles manipulés par la MT. On note dès maintenant que le code du symbole $\emptyset$ représentant le vide sera 0.

L'initialisation de la machine RAM consiste à lire le contenu initial du ruban de la MT et de le mémoriser dans les registres correspondants. Ceci

est fait par le morceau de programme suivant.

0 $C\sharp 1$ [R_1 est le premier registre pour le ruban]
1 $M0$ [Position de la tête de lecture]
2 *lire* [Lecture d'un symbole]
3 $\nrightarrow 9$ [Arrêt sur un $\emptyset$ codé par 0]
4 $M(0)$ [Mémorisation dans le registre correspondant]
5 $C0$
6 $\nearrow$ [Déplacement vers la droite]
7 $M0$
8 $\curvearrowright 2$
9 $C\sharp 1$ [Mise en place de la tête de lecture]
10 $M0$
11 $\curvearrowright q_0$ [Saut à l'état initial de la MT]

Ensuite, pour chacun des états de la MT, on écrit un morceau de programme qui simule des transitions de la MT telles que $q_0S_1S_2q_1D$; $q_0S_2S_3q_1G$ et $q_0\emptyset\emptyset q_0D$. Enfin, si un état q est final, le morceau de code de l'état q se contente d'écrire sur le ruban de sortie le contenu du ruban de la MT au moment où elle atteint l'état q.

Simulation d'une machine RAM par une MT

L'idée de la simulation est la suivante. La machine RAM est simulée par une MT à deux rubans. Le premier ruban contient le contenu de tous les registres de la machine RAM y compris celui de l'accumulateur. Les entiers contenus dans les registres sont écrits en binaire sur l'alphabet $\{0, 1\}$. Le contenu de l'accumulateur est mis en premier sur le ruban, suivi du contenu du registre R_1, puis du registre R_2 et ainsi de suite. Les écritures binaires des contenus sont séparées par le caractère $\ll , \gg$. A un instant donné, le ruban ne contient que les valeurs d'un nombre fini de registres. Le contenu des autres registres est représenté par leur valeur initiale qu'on suppose être zéro.

A chaque instruction du programme, on va associer une partie de la MT

qui aura pour fonction de simuler le fonctionnement de l'instruction donnée. Ces parties seront alors mises bout à bout pour constituer une MT qui simulera le programme en entier.

Les manipulations de registres nécessitent de savoir retrouver le contenu d'un registre. Pour ce faire, la MT écrit le numéro du registre sur le second ruban. Elle parcourt ensuite le premier ruban et à chaque lecture du symbole « , » elle augmente d'une unité l'entier inscrit sur le second ruban. Si la machine ne trouve pas assez de « , » sur le premier ruban, elle en ajoute en les séparant par 0 qui est le contenu initial d'un registre. Ensuite, les manipulations consistent à recopier le contenu du registre dans l'accumulateur pour les instructions *chargement* ou le contenu de l'accumulateur dans le registre pour l'instruction *mémorisation*. Dans le cas d'une indirection - adressage indirect - la machine copie d'abord le contenu du registre sur le second ruban puis recherche à nouveau le registre correspondant avant de copier les contenus.

Simuler les instructions arithmétiques est facile puisqu'il s'agit simplement de modifier le contenu de l'accumulateur qui se trouve au début du premier ruban.

Les instructions de saut seront quand à elles simulées *via* la façon dont les différentes parties seront combinées pour former la MT qui simulera le programme.

Annexe D

Indécidabilité du problème de l'arrêt propre aux MTA

La preuve qui suit a été fournie par Potgieter [Potgieter, 2006]. L'auteur démontre qu'une MTA - HM capable d'exécuter une infinité d'étapes de calcul en un temps fini - n'est pas capable de résoudre le problème de l'arrêt propre aux MTA. L'idée générale est de proposer une classe de machines qui ne peut pas résoudre le problème de l'arrêt propre à leur classe et de montrer que les MTA appartiennent à cette classe.

La preuve suppose que les symboles 1 et 0 sont des données en sortie valides pour toutes les machines considérées et que chaque machine X définit une fonction partielle ψ_X. Voici à présent la notion centrale de la preuve, à savoir la notion de *classe Chatrapur*.

Définition (Classe Chatrapur)
Une classe de machines $\mathcal{C}$ est une classe Chatrapur si

1. *Pour toute machine $X \in \mathcal{C}$ calculant une fonction partielle ψ_X, il existe un codage $X \to n_X$ où n_X peut servir comme donnée en entrée pour chaque machine de la classe $\mathcal{C}$.*
2. *Si $Y \in \mathcal{C}$ calcule une fonction totale ψ_Y dont les valeurs sont dans $\{0,1\}$, alors il existe une machine $X \in \mathcal{C}$ telle que $\psi_X \equiv \psi_Y|_{\psi_Y^{-1}(\{0\})}$.*

La première condition énonce - schématiquement - l'existence d'une machine programmable pouvant être universelle - au sens où elle pourrait simuler

le comportement de chaque machine de la classe. La seconde condition signifie quant à elle que ψ_X est identique à ψ_Y où $\psi_Y = 0$ et est non définie autrement. Continuons en définissant explicitement le problème de l'arrêt pour une classe de machines.

Définition (Problème de l'arrêt)
Le problème de l'arrêt pour une classe de machines $\mathcal{B}$ où chaque $X \in \mathcal{B}$ calcule une fonction partielle ψ_X, est résolu par f si

$$f(n_Z) = \chi_{\text{dom } \psi_Z}(n_Z) \quad \textit{pour tout } Z \in \mathcal{B}$$

Le problème de l'arrêt propre à une classe de machines $\mathcal{B}$ est de savoir s'il existe une machine $Y \in \mathcal{B}$ telle que ψ_X résoud le problème de l'arrêt pour $\mathcal{B}$.

Le théorème suivant montre que le problème de l'arrêt propre à la classe Chatrapur n'est pas résoluble.

Théorème
Il n'existe pas de classe Chatrapur dans laquelle le problème de l'arrêt est résoluble.

Preuve : Supposons que la machine $Y \in \mathcal{C}$ puisse résoudre le problème de l'arrêt pour la classe Chatrapur $\mathcal{C}$. Soit X la machine qui est définie relativement à Y *via* la seconde condition de la définition d'une classe Chatrapur et considérons $X(n_X)$. Puisque ψ_Y est une fonction totale, soit $\psi_Y(n_X) = 0$ soit $\psi_Y(n_X) = 1$. Dans le premier cas puisque Y résoud le problème de l'arrêt, $n_X \notin \text{dom } \psi_X$; ce qui est contradictoire puisque d'après la définition de X, ψ_X est définie sur $\psi_Y^{-1}(\{0\})$. Dans le second cas, si $\psi_Y(n_X) = 1$ alors $n_X \in \text{dom } \psi_X$ par définition de Y mais puisque $\psi_X = \psi_Y = 0$ sur dom ψ_X, on arrive aussi à une contradiction □

Pour montrer que le problème de l'arrêt propre aux MTA ne peut pas être résolu par ces mêmes machines, il ne reste plus qu'à prouver un dernier théorème.

Théorème
La classe $\mathcal{M}$ des MTA est une classe Chatrapur.

Preuve : Pour montrer que la classe $\mathcal{M}$ des MTA est une classe Chatrapur il suffit de montrer que cette classe satisfait les deux conditions énoncées dans la définition d'une classe Chatrapur. La satisfaction de la première condition est aisée à prouver car (1) la MTA est formellement décrite de la même manière qu'une MT - sauf en ce qui concerne le nombre d'étapes qu'elle peut exécuter ; et (2) une MT satisfait la première condition. Ensuite, pour qu'une MTA Y satisfasse la seconde condition il suffit de modifier légèrement son fonctionnnement de la manière suivante : si Y écrit 1 dans la première cellule de son ruban, elle active un ruban parallèle qui continue le calcul et boucle à l'infini - ce qui signifie que la fonction qui est calculée par la machine n'est pas définie pour la donnée en entrée sur le ruban. Si Y écrit 0 dans la première cellule, elle n'active pas le ruban parallèle et suit le cours normal de ses opérations □

Enfin on peut déduire des deux derniers théorèmes le corollaire suivant :

Corollaire 1

Le problème de l'arrêt propre aux MTA n'est pas résoluble par une MTA.

Annexe E

Concepts fondamentaux de la physique quantique

Je présente ici les concepts indispensables à la compréhension des processus quantiques et en particulier du modèle adiabatique proposé par Kieu - *cf.* chapitre 3. J'ai tenté autant que faire se peut de retranscrire le plus clairement possible l'ensemble de ces concepts afin de faciliter la compréhension du lecteur. Ces concepts sont néanmoins détaillés formellement dans le tome 1 de [Cohen-Tannoudji et al., 1973].

1 Les états quantiques

Un système quantique peut se trouver dans différents états quantiques. Un *état quantique* d'un système quantique est défini comme l'ensemble des caractéristiques physiques de ce système qui permet d'acquérir toute l'information possible sur le système. Les états quantiques sont représentés mathématiquement par des vecteurs. Cela veut dire que l'on peut représenter l'état d'un système par un objet mathématique -le vecteur - qui existe dans un espace mathématique - l'espace des états physiques du système. Puisque les vecteurs réprésentent des états quantiques, ces derniers sont appelés *vecteurs d'états*. Un système quantique est donc entièrement caractérisé par son vecteur d'état et toute l'information du système y est contenue.

2 Les valeurs propres et les vecteurs propres

Supposons que l'on dispose d'un système et d'un appareil de mesure, et que l'on exécute les opérations suivantes :

1. A l'instant t, on effectue une première mesure sur le système à l'aide de l'appareil de mesure. On trouve une valeur V_1.
2. On attend pendant un intervalle de temps Δt.
3. A l'instant $t + \Delta t$, on effectue une seconde mesure sur le système à l'aide du même appareil de mesure. On trouve une valeur V_2.

Intuitivement, on serait tenté de penser que si l'intervalle de temps Δt est suffisament petit - tellement petit que le système n'a pas eu le temps d'évoluer - alors la valeur V_2 trouvée lors de la seconde mesure devrait être la même que la valeur V_1 obtenue lors de la première mesure. Il se trouve que cette intuition est effectivement vérifiée expérimentalement : à la limite où l'intervalle de temps Δt entre les deux mesures tend vers 0, la valeur V_2 obtenue à la seconde mesure est la même que la valeur V_1 obtenue à la première mesure.

Ce simple résultat a de profondes implications en physique quantique. En effet, si l'appareil de mesure donne toujours le même nombre lors de la seconde mesure - si Δt tend vers 0 - c'est que le vecteur d'état qui caractérise l'état du système juste après la première mesure possède une propriété particulière : on est sûr, avec une probabilité égale à 1, de trouver un certain nombre - celui indiqué par la première mesure, en l'occurence V_1 - lors de la seconde mesure.

En général, le résultat d'une mesure n'est pas prévisible. Il existe toutefois certains états quantiques particuliers - appelés des *états propres* - pour lesquels le résultat de la mesure est absolument certain. Le résultat de la mesure est alors appelé *valeur propre*, et le résultat d'une mesure sur un système qui est dans un état propre est la valeur propre associée à cet état propre. Après une mesure sur un système, le système se trouve toujours dans un état propre.

D'un appareil de mesure à l'autre, les états propres peuvent être différents, c'est pourquoi on parle d'états propres relatifs à un appareil de mesure. Lorsqu'un système se trouve dans un état qui est un état propre, on dit que son état est un **vecteur propre**. Précisons à présent le sens mathématique de toutes ces notions.

3 Les observables

Une *observable* est un objet mathématique servant à caractériser les valeurs propres et les vecteurs propres d'un appareil de mesure. Une observable est une fonction mathématique dont l'ensemble de départ et l'ensemble d'arrivée est un espace de vecteurs - à savoir un espace vectoriel. Il prend donc un vecteur en entrée et donne un vecteur en sortie. En résumé, si O est une observable alors :

$$\begin{aligned} O &: \text{Espace vectoriel} \to \text{Espace vectoriel} \\ &: |\text{vecteur d'état}\rangle \longmapsto O|\text{vecteur d'état}\rangle \end{aligned} \tag{E.1}$$

A l'aide de ce nouvel objet, on peut caractériser mathématiquement les valeurs propres et les vecteurs propres. En effet, à chaque appareil de mesure on peut associer une observable. Un vecteur d'état non nul $|\text{état}\rangle$ est un état propre de l'observable O s'il existe un nombre λ tel que :

$$O|\text{état}\rangle = \lambda|\text{état}\rangle$$

c'est-à-dire si $O|\text{état}\rangle$ est proportionnel à $|\text{état}\rangle$. Le nombre λ et alors appelé *valeur propre de l'observable* O. Les vecteurs propres d'une observable sont donc certains vecteurs d'états particuliers.

4 L'observable d'énergie

L'*observable d'énergie* sera notée H et sera appelée *hamiltonien* du nom du physicien Hamilton. Si $|E\rangle$ est un état d'énergie bien définie E, c'est-à-dire

si c'est un état propre de l'observable H pour la valeur propre E, alors

$$H|E\rangle = E|E\rangle$$

En mécanique quantique, c'est l'énergie qui dicte l'évolution temporelle d'un système. Plus précisément, l'observable d'énergie H appliquée à un état est proportionnelle à sa variation dans le temps. Pour tout système dont l'état est décrit à chaque instant par le vecteur d'état $|\text{vecteur}\rangle(t)$ on a :

$$H|\text{vecteur}\rangle(t) = i\hbar\frac{d}{dt}|\text{vecteur}\rangle(t) \tag{E.2}$$

avec $i^2 = -1$ et $\hbar = \frac{h}{2\pi}$ où $h \simeq 6.610^{-34}$ $J.s$ est la constante de Planck. De plus, $\frac{d}{dt}$ signifie *dérivée par rapport au temps*. Enfin, pour les vecteurs quantiques on a par définition :

$$\frac{d}{dt}|\text{vecteur}\rangle(t) = \lim_{dt\to 0}\frac{|\text{vecteur}\rangle(t+dt) - |\text{vecteur}\rangle(t)}{dt}$$

L'équation (3.5) est appelée *équation de Schrödinger* et elle est valable tant que le système n'influe pas sur l'extérieur - tant qu'il n'y a pas de mesure. Si l'on connaît tous les états propres de l'énergie, il est alors possible de résoudre explicitement cette équation.

Index

Bibliographie

S. Aaronson. NP-Complete Problems and PhysicalReality. *arXiv :quant-ph/0502072*, 2005.

A.A Abbott, C.S. Calude, J. Conder, and K. Svozil. Strong Kochen-Specker Theorem and Uncomputability of Quantum Randomness. *Physical Review*, 86 :1–11, 2012.

W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99, pp. 118-133, 1928.

L.M. Adleman and M. Blum. Inductive Inference and Unsolvability. *The Journal of Symbolic Logic*, (3), 56, pp. 891-900, 1991.

V. Allis and T. Koetsier. On Some Paradoxes of the Infinite II. *British Journal for the Philosophy of Science*, 42, pp. 187-194, 1991.

V. Allis and T. Koetsier. On Some Paradoxes of the Infinite. *British Journal for the Philosophy of Science*, 46, (2), pp. 235-247, 1995.

H. Andréka and I. Németi. General Relativistic Hypercomputing and Foundations of Mathematics. *Natural Computing*, 8, pp. 499-516, 2009.

D. Angluin and C Smith. Inductive Inference : Theory and Method. *Computing Survey*, 15, pp. 237-269, 1983.

V. Ardourel. *?* PhD thesis, Universit´é Paris 1 Panthéon-Sorbonne, 2013.

J.V. Atanasoff. Computing Machine for the Solution of Large Systems of Linear Algebraic Equations. Technical report, Ames, IA : Iowa State College, 1940.

J.L Balcàzar, J. Dìaz, and J. Cabarrò. *Structural Complexity*, volume 1. Berlin : Springer, 1995.

J.D. Barrow. Wigner Inequalities for a Black Hole. *Physical Review*, 54D, pp. 6563-6564, 1996.

K.J. Barwise. Infinitary Logic and Admissible Sets. *The Journal of Symbolic Logic*, (2), 34, pp. 226-252, 1969.

K.J. Barwise. Mathematical Proofs of Computer System Correctness. *Notices of the American Mathematical Society*, Vol. 36, pp. 844-851, 1989.

A. Bell and M. Holcombe. Computational Models of Cellular Processing. In M. Holcombe, R. Paton, and R. Cuthbertson, editors, *Computation and Molecular Biological Systems*. Singapore : World Scientific Press, 1996.

J.L. Bell. Infinitary Logic. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2006.

P. Benacerraf. Super-Tasks and the Modern Eleatics. *Journal of Philosophy*, 59 pp. 765-785, 1962.

P. Benacerraf. God, the Devil, and Gödel. *Monist*, 51 pp. 9-32, 1967.

P. Benioff. Quantum Mechanical Hamiltonien Models of Turing Machines. *Journal of Statistical Physics*, 29, pp. 515, 1982.

C. Bennett. Physical Limits to Computation. *IBM Journal Research Development*, 58, 1985.

C. Bennett. Notes on the History of Reversible Computation. *IBM Journal Reasearch Development*, 32, pp. 16-23, 1988.

Charles Bennett. Logical Reversibility of Computation. *IBM Journal Research Development*, 10, 1973.

R. Black. Proving Church's Thesis. *Philosophia Mathematica*, (3), Vol. 8, pp. 244-258, 2000.

R.M. Blake. The Paradox of Temporal Process. *Journal of Philosophy*, 23, pp. 645-654, 1926.

R. Blanché. *L'Axiomatique*. Presse Universitaire de France, 1999.

L. Blum, M. Shub, and S. Smale. On a Theory of Computation and Complexity over the Real Numbers : NP-Completeness, Recursive Functions and Universal Machines. *Bulletin of the Mathematical American Society*, 21, (1), pp. 645-654, 1926.

G. Boolos and R.C. Jeffrey. *Computability and Logic*. Cambridge : Cambridge University Press, 2005.

O. Bournez. A Survey on Continuous Time Computations. In S.B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms. Changing Conceptions of What is Computable*. Springer-Verlag, 2008.

G. L. Bowie. An Argument Against Church's Thesis. *The Journal of Philosophy*, (3), Vol. 70, pp. 66-76, 1973.

R.S. Boyer and J.S. Moore. Proof-Checking, Theorem-Proving and Program Verification. In W.W. Bledsoe and D.W. Loveland, editors, *Automated Theorem Proving : After 25 Years*, volume 29 of *Contemporary Mathematics*, pages 119–132. Providence : RI, 1984.

P. Braconnot and O. Marti. La Modélisation du Climat. *Clefs CEA*, 47 : 16–22, 2002.

S. Bringsjord. An Argument for the Uncompatibility of Infinitary Mathematical Expertise. In P. Hayes P. Feltovich, K. Ford, editor, *Expertise in Context*, pages 475–497. AAAI Press : Menlo Park, CA, 1997.

S. Bringsjord and K. Arkoudas. The Modal Argument for Hypercomputing Minds. *Theoretical Computer Science*, 317 pp. 167-190, 2004.

S. Bringsjord and M. Zenzen. *Superminds : People Harness Hypercomputation, and More*. Kluwer Academic Publishers, Dordrecht, the Netherlands, 2003.

S. Bringsjord, O. Kellett, A. Shilliday, J. Taylor, B. van Heuveln, Y. Yang, J. Baumes, and K. Ross. A New Gödelian Argument for Hypercomputing Minds based on the Busy Beaver Problem. In *Applied Mathematics and Computation*, 2011.

A.R. Burks. *Who Invented the Computer ?* Amherst, MA : Prometheus, 2002.

A.R. Burks and A.W. Burks. *The First Electronic Computer : The Atanasoff Story of Computing*. Ann Arbor : University of Michigan Press, 1988.

J. Byl. On resolving the Littlewood-Ross paradox. *Missouri Journal of Mathematical Sciences*, 12, pp. 42-47, 2000.

C.S. Calude. Who is Afraid of Randomness ? *CDMTCS Research Report Series*, 143, pp. 1-15, 2000.

C.S. Calude. Algorithmic randomness, quantum physics, and incompleteness. In M. Margenstern, editor, *Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2005.

C.S. Calude and G. Păun. Bio-Steps Beyond Turing. *Biosystems*, 77 :175–194, 2004.

C.S. Calude and K. Svozil. Quantum Randomness and Value Indefiniteness. *Advanced Science Letters*, 1, pp. 65-68, 2008.

G.J. Chaitin. On the Length of Programs for Computing Finite Binary Sequences. *Journal of the Association for Computing Machinery*, 13, pp. 547-569, 1966.

G.J. Chaitin. Algorithmic Information Theory. *IBM Journal of Research and Development*, 31, pp. 350-359, 1977.

G.J. Chaitin. *Information, Randomness and Incompleteness : Papers on Algorithmic Information Theory*. Singapore : World Scientific, 1987.

D. Chalmers. On Implementing a Computation. *Minds and Machines*, 4, pp. 391-402, 1995.

D. Chalmers. Does a Rock Implement Every Finite-State Automaton ? *Synthese*, 108, pp. 335-359, 1996.

C.S. Chihara. On the Possibility of Completing an Infinite Task. *Philosophical Review*, 74, pp. 74-87, 1965.

R.L. Chrisley. Why Everything Doesn't Realize Every Computation. *Minds and Machines*, 4, pp. 403-430, 1995.

I.L. Chuang and M.A. Nielsen. *Quantum Computation and Quantum Information*. Cambridge : Cambridge University Press, 2000.

A. Church. An Unsolvable Problem of Elementary Number Theory. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

A. Church. Reviews of Turing 1936-7 and Post 1936. *Journal of Symbolic Logic*, Vol. 2, pp. 42-43, 1937.

A. Church. On the Concept of a Random Sequence. *Bulletin of the American Mathematical Society*, 46, pp. 130-135, 1940.

P. Churchland and P.S. Churchland. Stalking the Wild Epistemic Engine. *Noûs*, XVII pp. 5-18, 1983.

P.S. Churchland and T.J. Sejnowski. *The Computational Brain*. Cambridge MA : MIT Press, 1992.

C.E. Cleland. On Effective Procedures. *Minds and Machines*, 12, pp. 159-179, 2002.

C.E. Cleland. The Concept of Computability. *Theoretical Computer Science*, 317, pp. 209-225, 2004.

I.B. Cohen. *Howard Aiken : Portrait of a Computer Pioneer*. Cambridge MA : MIT Press, 1999.

C. Cohen-Tannoudji, B. Dui, and F. Lalöe. *Mécanique Quantique*. Paris : Hermann, 1973.

S.A. Cook. An Overview of Computational Complexity. *Communication of the ACM*, 26, (6), pp. 151-158, 1983.

S.B. Cooper. *Computability Theory*. Chapman Hall/CRC Mathematics Series, 2004.

S.B. Cooper. What Makes a Computation Unconventional ? In G. Dodig-Crnkovic and R. Giovagnoli, editors, *Computing Nature*. Berlin : Springer-Verlag, 2013.

A.H. Copeland and P. Erdös. Note on Normal Numbers. *Bulletin of the American Mathematical Society*, 52, pp. 857-860, 1946.

B.J. Copeland. What is Computation ? *Synthese*, (3), 108, pp. 335-359, 1996.

B.J. Copeland. The Broad Conception of Computation. *American Behavorial Scientist*, 40, pp. 690-716, 1997.

B.J. Copeland. Narrow Versus Wide Mechanism : Including a Re-Examination of Turing's Views of the Mind-Machine Issue. *The Journal of Philosophy*, 1, pp. 5-32, 2000a.

B.J. Copeland. The Modern History of Computing. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2000b.

B.J. Copeland. The Church-Turing Thesis. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2002a.

B.J. Copeland. Hypercomputation. *Minds and Machines*, 12, pp. 461-502, 2002b.

B.J. Copeland. Accelerating Turing Machine. *Minds and Machines*, 12, pp. 281-301, 2002c.

B.J. Copeland. Hypercomputation : Philosophical Issues. *Theoretical Computer Science*, 317, pp. 251-267, 2004.

B.J. Copeland and D. Proudfoot. Alan Turing's Forgotten Ideas in Computer Science. *Scientific American*, 208, pp. 76-81, 1999.

B.J. Copeland and O. Shagrir. Physical Computation : How General are Gandy's Principles for Mechanisms ? *Minds and Machines*, (2), 17, pp. 217-231, 2007.

B.J. Copeland and O. Shagrir. Do Accelerating Turing Machines Compute the Uncomputable ? *Minds and Machines*, 21, pp. 217 - 231, 2011.

P. Cotogno. Hypercomputation and the Physical Church-Turing Thesis. *British Society for the Philosophy of Science*, 54, pp. 181-223, 2003.

P. Cotogno. A Brief Critique of Pure Hypercomputation. *Minds and Machines*, 19, pp. 391-405, 2009.

R. Cummins. *The Nature of Psychological Explanation.* Cambridge, MA : MIT Press, 1983.

O.J. Dahl, E.W. Dijkstra, and C.A.R Hoare. *Strutured Programming.* London : Academic Press, 1972.

B. Davies. Building Infinite Machines. *British Journal for the Philosophy of Science*, 52, pp. 671-682, 2001.

M. Davis. *Computability and Unsolvability.* Mineola, New York : Dover, 1958.

M. Davis. The Myth of Hypercomputation. In C. Teuscher, editor, *Alan Turing : Life and Legacy of a great Thinker*, pages 195–212. Berlin : Springer, 2004.

M. Davis. Computability, Computation, and the Real World. In S. Termini, editor, *Imagination and Rigor*, pages 63–70. Milan : Springer, 2006.

M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 3, pp. 201-215, 1960.

K. De Leeuw, E.F. Moore, C.E. Shannon, and N. Shapiro. *Computability by Probabilistic Machines*. Automata Studies : Princeton University Press, 1956.

J.P. Delahaye. Randomness, Unpredictability and Absence of Order. In Jacques Dubucs, editor, *Philosophy of Probability*, pages 145–167. Dordrecht : Kluwer, 1993.

J.P. Delahaye. *Le Fascinant Nombre* π. Belin, 1997.

J.P. Delahaye. *L'Intelligence et le Calcul*. Belin, 2002.

J.P. Delahaye. Les Chiffres de la Complexité Informatique. *Pour la Science*, 314, pp. 162-167, 2003.

J.P. Delahaye. *Complexités : aux Limites des Mathématiques et de l'Informatique*. Belin, 2006.

J.P. Delahaye. The Martin-Löf-Chaitin Thesis : the Identification by Recursion Theory of the Mathematical Notion of Random Sequence. In Hector Zénil, editor, *Randomness Through Computation*, pages 121–140. Singapore : World Scientific Association, 2011.

H. Delfs and H. Knebl. *Introduction to Cryptography : Principles and Applications*. Information Security and Cryptography. Springer, 2007.

N. Dershowitz and Y. Gurevich. A Natural Axiomatisation of Computability and Proof of Church's Thesis. *The Bulletin of Symbolic Logic*, 14(3) :299–350, 2008.

D. Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London*, A 400, pp. 97-117, 1985.

K. Douglas. Super-Turing Computation : A Case Study Analysis. Master's thesis, Carnegie Mellon University, available at http ://www.philosopher-animal.com/papers/take6c.PDF, 2003.

J. Dubucs. *Recherches sur la Théorie de la Démonstration.* PhD thesis, Université Paris 1 Panthéon-Sorbonne, 1984.

J. Dubucs. Feasibility in Logic. *Synthese*, 132, pp. 213-237, 2002.

R.J. Duffin. Rubel's Universal Differential Equation. *Proceedings of the National Academy of Sciences USA 78*, 8, [Part 1 : Physical Sciences], pp. 4661-4662, 1981.

J. Durand-Lose. Abstract Geometrical Computation 5 : Embedding Computable Analysis. *Natural Computing*, 10(4) :1261–1273, 2011.

A. Duwell. *How to Teach an Old Dog New Tricks : Quantum Information, Quantum Computation, and the Philosophy of Physics.* PhD thesis, University of Pittsburgh, 2004.

J. Earman. *Bangs, Crunchs, Whimpers and Shrieks - Singularities and Acausalities in Relativistic Spacetimes.* Oxford : Oxford University Press, 1995.

J. Earman and J.D. Norton. Forever is a Day : Supertasks in Pitowski and Malament-Horgarth Spacetimes. *Philosophy of Science*, 60 pp. 22-42, 1993.

J. Earman and J.D. Norton. Infinite Pains : the Trouble with Supertasks. In A. Morton and S.P. Stich, editors, *Paul Benacerraf : The Philosopher and His Critics.* Cambridge, MA : Wiley-Blackwell, 1996.

H.D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic.* New York, NY, Springer-Verlag, 1984.

S. Eilenberg. *Automata, Languages and Machines.* Academic Press, 1974.

G. Etesi and I. Németi. Non-Turing Computations Via Malament-Hogarth Space-Times. *International Journal of Theoretical Physics*, (2), 41, pp. 341-370, 2002.

E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum Computation by Adiabatic Evolution. *http ://arxiv.org/quant-ph/0001106*, 2000.

S. Feferman. Gödel's Incompleteness Theorems, Free Will and Mathematical Thought. In R. Swinburn, editor, *Free Will and Modern Science*. British Academy Publications Online, 2011.

R.P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21, pp. 467-488, 1982.

J. Fodor. *The Language of Thought*. Cambridge, MA : Harvard University Press, 1975.

J. Fodor. The Mind-Body Problem. *Scientific American*, CCXLIV, pp. 124-132, 1981.

J. Folina. Church's Thesis : Prelude to a Proof. *Philosophia Mathematica*, (3), Vol. 6, pp. 302-323, 1998.

F. Franchette. Are Mathematical Models of Hypercomputation Useful ? In Ess and R. Hagengruber, editors, *The Computational Turn : Past, Present, Future*, pages 46–49. IACAP Conference Proceedings (International Association of Computing and Philosophy), The University of Aarhus, 2011.

F. Franchette. La Thèse de l'Hyper-Calcul : Problèmes et Enjeux Philosophiques. *Cahier thématique de Philosophia Scientiae L'année Alan Turing*, 16/3, pp. 17-38, 2012.

F. Franchette. Oracle Hypermachines Faced with the Verification Problem. In G. Dodig-Crnkovic and R. Giovagnoli, editors, *Computing Nature*. Berlin : Springer-Verlag, 2013.

T. Franzen. *Gödel's Theorem. A Incomplete Guide to Use and Abuse*. Wellesley, MA : A.K. Peters, 2005.

E. Fredkin and T. Toffoli. Conservative Logic. *International Journal of Theoretical Physics*, 21, pp. 219-253, 1982.

P. Gacs. Every Sequence is Reductible to a Random One. *Information and Control*, 70, pp. 186-192, 1986.

R. Gandy. Church's Thesis and Principles for Mechanisms. In J. Barwise, H.J. Keisler, and K. Kunen, editors, *The Kleene Symposium : Proceedings of the Symposium Held June 18-24, 1978 at Madison, Winconsin, USA*. North-Holland Publishing Company Amsterdam, 1980.

R. Gandy. The Confluence of Ideas in 1936. In R. Herken, editor, *The Universal Turing Machine*. Oxford : Oxford University Press, 1988.

K. Gödel. On Formally Undecidable Propositions of the Principia Mathematica and Related Systems, I. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1931.

K. Gödel. On Undecidable Propositions of Formal Mathematical Systems. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1934.

K. Gödel. Remarks before the Princeton Bicentennial Conference on Problems in Mathematics. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1946.

K. Gödel. Postscriptum (Gödel 1934). In S. Feferman, JR. J.W. Dawson, S.C. Kleene, G.H. Moore, R.M. Solovay, and J. van Heijennoort, editors, *Kurt Gödel, Collected Works, Vol. I : Publications 1929-1936 (1986)*. New York : Oxford University Press, 1965.

K. Gödel. *Kurt Gödel, Collected Works, Vol. I : Publications 1929-1936*. New York : Oxford University Press, 1986.

G. Gentzen. *The Collected Papers of Gerhard Gentzen*. North-Holland Pub.Co., 1969.

P. Godfrey-Smith. Triviality Arguments Against Functionalism. *Philosophical Studies*, (2), 145, pp. 273-295, 2009.

M. Gold. Limiting Recursion. *The Journal of Symbolic Logic*, 30, pp. 28-48, 1965.

M. Gold. Language Identification in the Limit. *Information and Control*, (5), 10, pp. 447-474, 1967.

M.B. Green, J.H. Schwarz, and E. Witten. *Superstring Theory*. Cambridge : Cambridge University Press, 1987.

L.K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *28th Annual ACM Symposium on the Theory of Computing*, May 1996.

A. Hagar and A. Korolev. Quantum Hypercomputability ? *Minds and Machines*, (1), 16, pp. 87-93, 2006.

A. Hagar and A. Korolev. Quantum Hypercomputation : Hype or Computation ? *Philosophy of Science*, (3), 74, pp. 347-363, 2007.

J.D. Hamkins. Infinite Time Turing Machines. *Minds and Machines*, 12 (4) pp. 521-539, 2002.

J.D. Hamkins and A. Lewis. Infinite time turing machines. *The Journal of Symbolic Logic*, 65 (2) pp. 567-604, 2000.

S. W. Hawking. Particle Creation by Black Holes. *Communications in Mathematical Physics*, 43, pp. 199-220, 1975.

D. Hilbert. Uber das Unendliche. *Mathematische Annalen*, 95, pp. 161-190, 1926. English translation in van Heijenoort (1967, 367-392.

A. Hodges. Can Quantum Computing Solve Classically Unsovable Problems ? *arXiv :quant-ph/0512248*, 2005.

M. Hogarth. Does General Relativity Allow an Observer to View an Eternity in a Finite Time ? *Foundations of Physics Letters*, 5, pp. 173-181, 1992.

M. Hogarth. Non-Turing Computers and Non-Turing Computability. *PSA : Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1, Contrubuted Papers (1994), pp. 126-138, 1994.

M. Hogarth. Deciding Arithmetic Using SAD Computers. *The British Journal for the Philosophy of Science*, (4), 55, pp. 681-691, 2004.

J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

E. Horowitz, S. Sahni, and S. Rajasekaran. *Computer Algorithms*. Silicon Pr, 2 edition, 2007.

S. Horst. The Computational Theory of Mind. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2003.

N. Huggett. Zeno's Paradoxes. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2010.

P. Humphreys. *Extending Ourselves : Computational Science, Empiricism, and Scientific Method*. New York : Oxford University Press, 2004.

C. Imbert. *L'opacité Intrinsèque de la Nature : Théorie Connues, Phénomènes Difficiles à Expliquer et Limites de la Science*. PhD thesis, Université Paris 1 Panthéon-Sorbonne, 2008.

A.S. Jackson. *Analog Computation*. New York : McGraw-Hill, 1960.

T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger. A Fast and Compact Quantum Random Number Generator. *Review of Scientific Instruments*, 71, pp. 1675-1680, 2000.

C.L. Johnson. *Analog Computer Techniques*. New York : McGraw-Hill, second edition, 1963.

M.W. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, N. Dickson, F. Hamze, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum Annealing with Manufactured Spins. *Nature*, (473) :194–198, 2011.

J.A. Jones and M. Mosca. Implementation of a Quantum Algorithm on a Nuclear Magnetic Resonance Quantum Computer. *Journal of Chemical Physics*, 109, pp. 1648-1654, 1998.

L. Kalmar. An Argument against the Plausibility of Church's Thesis. In A. Heyting, editor, *Constructivity in Mathematics*. Amsterdam : North-Holland, 1959.

E. Kamke. Über neuere Begründungen der Wahrscheinlichkeitsrechnung. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 42, pp. 14-27, 1932.

R.M. Karp and R.J. Lipton. Some Connections Between Nonuniform and Uniform Complexity Classes. *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, pp. 302-309, 1980.

T. Kato. On the Adiabatic Theorem of Quantum Mechanics. *Journal of the Physical Society of Japan*, 5, pp. 435-439, 1950.

H.J. Keisler. *Model Theory for Infinitary Logic*. Amsterdam, the Netherlands, North-Holland, 1971.

T. Kieu. Quantum Algorithm for the Hilbert's Tenth Problem. *Archiv*, quant-ph/0110136, 2001.

T. Kieu. Quantum Hypercomputability. *Minds and Machines*, 12 (4) pp. 541-561, 2002.

T. Kieu. Computing the non-Computable. *Contemporary Physics*, (1), 44, pp. 541-561, 2003.

T. Kieu. A Reformulation of Hilbert's Tenth Problem trough Quantum Mechanics. *Proceedings of the Royal Society*, A460, pp. 1535-1545, 2004.

T. Kieu. A Anatomy of a Quantum Adiabatic Algorithm that Transcends the Turing Computability. *International of Quantum Information*, (1), 3, pp. 177-183, 2005.

T. Kieu. Quantum Adiabatic Algorithm for Hilbert's Tenth Problem. *http ://arxiv.org/quant-ph/0310052v2*, 2008.

T. Kieu and A. Buchdahl. Hypercomputability of Quantum Adiabatic Processes : Facts versus Prejudices. *http ://arxiv.org/quant-ph/0504101*, 2005.

T. Kieu and T. Ord. The Diagonal Method and Hypercomputation. *The British Journal for the Philosophy of Science*, 56 pp. 147-156, 2005.

D. King. Is the Mind a Turing Machine ? *Synthese*, (3), Vol. 108, pp. 379-389, 1996.

S.C. Kleene. General Recursive Functions of Natural Numbers. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

S.C. Kleene. Recursive Predicates and Quantifiers. *Transactions of the American Mathematical Society*, 53, pp. 41-75, 1943.

S.C. Kleene. *Mathematical Logic*. John Wiley & Sons, Inc., New York, London, Sydney, 1967.

S.C. Kleene. Origins of Recursive Function Theory. *Annals of the History of Computing*, 3, pp. 52-57, 1981.

C. Klein. Dispositional Implementation Solves the Superfluous Structure Problem. *Synthese*, 165, pp. 141-153, 2008.

D.E. Knuth. *Art of Computer Programming, Volume 2 : Seminumerical Algorithms*. Addison Wesley, 3 edition, 1998.

A.N. Kolmogorov. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*, 1, (1), pp. 1-7, 1965.

A.N. Kolmogorov and V.A. Uspensky. On the Definition of an Algorithm. *Uspekhi Mat. Naut*, 13, (4), 1958. English translation in AMS translation vol. 21 (1963), pp. 217-245.

A.N. Kolmogorov and V.A. Uspensky. Algorithms and Randomness. *SIAM Theory Probability Applied*, 32, pp. 389-412, 1987.

G. Kreisel. A Notion of Mechanistic Theory. *Synthese*, 29, pp. 11-26, 1974.

S. Kripke. *Wittgenstein on Rules and Private Language : an Elementary Exposition.* Cambridge, MA : Harvard University Press, 1982.

J.L. Krivine. *Théorie des Ensembles.* Nouvelle bibliothèque math'ematique. Cassini, 2007.

R.P. Kurshan. Program Verification. *Notices of the AMS*, (5), 47, pp. 534-545, 2000.

R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal Research Development*, 44, pp. 261-269, 1961.

T.K. Landauer. How Much do People Remember ? Some Estimates of the Quantity of Learned Information in Long-Term Memory. *Cognitive Science*, 10, (4), pp. 477-493, 1986.

J.P. Laraudogoitia. A Beautiful Supertask. *Mind*, 105, pp. 81-83, 1996.

J.P. Laraudogoitia. Infinity Machines and Creation Ex Nihilo. *Synthese*, 115, pp. 259-265, 1998.

I. Lavallée. *Complexité et Algorithmique Avancée.* Hermann, 2008.

G. Laycock. *The Theory and Practice of Specification Based Software Testing.* PhD thesis, Department of Computer Science, University of Sheffield, UK, 1993.

M. Le Bellac. *Introduction à l'Information Quantique.* Belin, 2008.

L.A. Levin. Randomness Conservative Inequalities : Information and Independance in Mathematical Theories. *Information and Control*, 61, pp. 15-37, 1984.

L.A. Levin. Average Case Complete Problems. *SIAM Journal on Computing*, 15, pp. 285-286, 1986.

D. Lewis. Lucas Against Mechanism. *Philosophy*, 44, pp. 231-233, 1969.

P. Lindström. Penrose's New Argument. *Journal of Philosophical Logic*, 30, pp. 241-250, 2001.

P. Lindström. Remarks on Penrose's New Argument. *Journal of Philosophical Logic*, 35, pp. 231-237, 2006.

J.E. Littlewood. *Littlewood's* M*iscellany*. Cambridge University Press, 1986.

S. Lloyd. Ultimate Physical Limits to Computation. *Nature*, 406, pp. 1047-1054, 2000.

B. Loff and J.F. Costa. Five Views of Hypercomputation. *International Journal of Unconventional Computing*, 5, pp. 193-207, 2009.

J.R. Lucas. Minds, Machines and Gödel. *Philosophy*, 36, pp. 112-127, 1961.

J.R. Lucas. Mechanism : a Rejoinder. *Philosophy*, 45, pp. 149-151, 1970.

J.R. Lucas. This Gödel is Killing me : a Rejoinder. *Philosophia*, (6), pp. 145-148, 1976.

J.R. Lucas. Mind, Machines and Gödel : a Retrospect, April 1990. a Paper Read to the Turing Conference at Brighton.

J.R. Lucas. The Godelian Argument, 1996. a Paper Read to the BSPS in Oxford.

W. Lycan. Form, Function and Feel. *Journal of Philosophy*, (1), 78, pp. 24-50, 1981.

P. Lynds. Zeno's Paradoxes : a Timely Solution. 2003.

J. Machta. Entropy, Information, and Computation. *American Journal of Physics*, 67(12) :1074–1077, 1999.

Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill : Dover, 1974.

O. Maroney. Information Processing and Thermodynamic Entropy. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2009.

P. Martin-Löf. The Definition of a Random Sequence. *Information and Control*, 9, pp. 602-619, 1966.

Y. Matiiassevitch. *Le Dixième Problème de Hilbert : son Indécidabilité*. Masson, 1995.

T. Maudlin. Computation and Consciousness. *Journal of Philosophy*, (8), 86, pp. 407-432, 1989.

J.C. Maxwell. *Theory of Heat*. New York : Longmans Green, 1908.

S. McCall. Can a Turing Machine Know that the Gödel Sentence is True ? *The Journal of Philosophy*, 96, pp. 525-532, 1999.

W.I. McLaughlin. Resolving Zeno's Paradoxes. *Scientific American*, November 1994.

D. McMahon. *Quantum Computing Explained*. Wiley Inter-Science, 2008.

E. Mendelson. Second Thoughts about Church's Thesis. *The Journal of Philosophy*, (5), Vol. 87, pp. 225-233, 1990.

R.C. Merkle. Energy Limits to the Computational Power of The Human Brain. *Foresight Update*, 6, http ://www.merkle.com/brainLimits.html, 1989.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin : Springer-Verlag, second edition, 2002.

J.W. Mills. The Nature of the Extended Analog Computer. *Physica D : Nonlinear Phenomena*, 237, pp. 1235-1256, 2008.

C. Moore. Unpredictability and Undecidability in Dynamical Systems. *Physical Review Letters*, 64, pp. 2354-2357, 1990.

G. Moore. Cramming more Components onto Integrated Circuits. *Electronics*, 38, (8), pp., 1965.

R. Morris. *Achilles and the Quantum Universe.* Redwoods Books, Trowbridge : Wiltshire, 1997.

P. Morrison and E. Morrison. *Charles Babbage and his Calculating Engines.* New York : Dover, 1961.

J. Mycka, J.F. Costa, and F. Coelho. The Euclid Abstract Machine. *International Journal of Unconventional Computing*, 4, (3), pp. 223-248, 2008.

E. Nagel and J. Newman. *Gödel's Proof.* New York University Press, 2008.

G.J. Nam, K.A. Sakallah, and R.A. Rutenbar. A New FPGA Detailed Routing Approach via Search-Based Boolean Satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (6), 21, pp. 674, 2002.

I. Németi and G. Dàvid. Relativistic Computers and the Turing Barrier. *Journal of Applied Mathematics and Computation*, 178, pp. 118-142, 2006.

A. Newell. Physical Symbols Systems. *Cognitive Science*, Vol. 4, pp. 135-183, 1980.

P. Odifreddi. *Classical Recursion Theory.* Amsterdam : Elsevier, 1989.

T. Ord. Hypercomputation : Computing more than the Turing Machine. Technical report, University of Melbourne, Melbourne, Australia, arXiv :math.LO/0209332, 2002.

C.H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

R. Penrose. Twistor Algebra. *Journal of Mathematical Physics*, (2), 8, pp. 345-366, 1967.

R. Penrose. *The Emperor's New Mind : Concerning Computers, Minds, and the Laws of Physics.* Oxford, UK : Oxford University Press, 1989.

R. Penrose. *Shadows of the Mind.* Oxford, UK : Oxford University Press, 1994.

R. Penrose. *The Road of Reality : a Complete Guide to the Laws of the Universe*. New York, USA : Alfred A. Knopf, 2005.

R. Peter. Rekursivitat und Konstruktivitat. In A. Heyting, editor, *Constructivity in Mathematics*. Amsterdam : North-Holland, 1959.

G. Piccinini. Computing Mechanisms. *Philosophy of Science*, 74, pp. 501-526, 2007.

G. Piccinini. Computers. *Pacific Philosophical Quaterly*, 89, pp. 32-73, 2008a.

G. Piccinini. Some Neural Networks Compute, Other's Don't. *Neural Networks*, 21, pp. 311-321, 2008b.

G. Piccinini. Computation in Physical Systems. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2010.

G. Piccinini. The Physical Church-Turing Thesis : Modest or Bold ? *The British Journal For The Philosophy of Science*, 62, pp. 773-769, 2011.

I. Pitowsky. The Pysical Church Thesis and Physical Computational Complexity. *Iyyun*, 39 pp. 81-99, 1990.

E.L. Post. Finite Combinatory Processes. Formulation I. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

E.L. Post. Absolutely Unsolvable Problems and Relatively Undecidable Propositions- Account of an Anticipation. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1941.

P. Potgieter. Zeno Machines and Hypercomputation. *Theoretical Computer Science*, (1), vol. 358, pp. 23-33, 2006.

M.B. Pour-El. Abstract Computability and its Relation to the General Purpose Analog Computer (Some Connections between Logic, Differential Equations and Analog Computers. *Transactions of the American Mathematical Society*, 199, pp. 1-28, 1974.

M.B Pour-El and J. Richard. The Wave Equation with Computable Initial Data such as its Unique Solution is not Computable. *Advances in Mathematics*, 39, pp. 215-39, 1981.

H. Putnam. Minds and Machines. In S. Hook, editor, *Dimensions of Mind : A Symposium*. New York : Collier, 1960.

H. Putnam. Trial and Error Predicates and the Solution to a Problem of Motowski. *The Journal of Symbolic Logic*, 30, (1), pp. 49-57, 1965.

H. Putnam. Psychological Predicates. In W.H. Capitan and DD. Merrill, editors, *Art, Mind, and Religion*. Pittsburgh, PA : University of Pittsburgh Press, 1967.

H. Putnam. *Philosophical Papers : Volume 2, Mind, Language and Reality*. Cambridge : Cambridge University Press, 1975.

H. Putnam. *Representation and Reality*. Cambridge, MA : MIT Press, 1988.

G. Păun. *Membrane Computing*. Berlin : Springer-Verlag, 2002.

G. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing*. Berlin : Springer-Verlag, 1998.

Z.W. Pylyshyn. *Computation and Cognition*. Cambridge, MA : MIT Press, 1984.

M. Redhead. Mathematics and the Mind. *The British Journal for the Philosophy of Science*, 55, 731-737, 2004.

H. Rogers. *Theory of Recursive Functions and Effective Computability*. The MIT Press, 1987.

R. Rojas and U. Hashagen. *The First Computers-History and Architectures*. Cambridge, MA : MIT Press, 2000.

D. Ross. Church's Thesis : What its Difficulties Are and Are not. *The Journal of Philosophy*, (15), Vol. 71, pp. 515-525, 1974.

S. Ross. *A First Course in Probability*. New York and London : Macmillian, 3rd edition, 1988.

C. Rovelli. Loop Quantum Gravity. *Living Reviews in Relativity*, (1, (1), pp. 1-75), 1998. URL `http://www.livingreviews.org/lrr-1998-1`.

C. Rovelli. *Quantum Gravity*. Cambridge : Cambridge University Press, 2004.

L.A. Rubel. The Extended of the Analog Computer. *Advances in Applied Mathematics*, 14, pp. 39-50, 1993.

B. Russell, A. Urquhart, and A.C. Lewis. *Foundations of Logic : 1903-05*. Selections. Routledge, 1994.

W.C. Salmon. *Zeno's Paradoxes*. Indianapolis : Hackett Pub. Co. Inc., 2001.

M. Scheutz. When Physical Systems Realize Functions ...,. *Minds and Machines*, (2), 9, pp. 161-196, 1999.

M. Scheutz. Causal versus Computational Complexity. *Minds and Machines*, (4), 11, pp. 534-556, 2001.

C.P. Schnorr. A Unified Approach to the Definition of a Random Sequence. *Mathematical System Theory*, (3), 5, pp. 246-258, 1971.

C.P. Schnorr. A Survey of the Theory of Random Sequences. In D. Reidel R.E. Butts, J. Hintikka, editor, *Basic Problems in Methodology and Linguistics*. Dordrecht, 1977.

M. Scriven. The Mechanical Concept of Mind. *Mind*, 62 :230–240, 1953.

J. Searle. *The Rediscovery of the Mind*. Cambridge : MIT, 1992.

J. Searle. *The Mystery of Consciousness*. New York : New York Review of Books, 1997.

O. Shagrir. Why we View the Brain as a Computer. *Synthese*, (3), 153, pp. 396-416, 2006.

O. Shagrir and I. Pitowsky. Physical Hypercomputation and the Church-Turing Thesis. *Minds and Machines*, 13 pp. 87-101, 2003.

C.E. Shannon. Mathematical Theory of the Differential Analyser. *Journal of Mathematics and Physics*, XX, pp. 337-354, 1941.

S. Shapiro. Mechanism, Truth, and Penrose's New Argument. *Journal of Philosophical Logic*, 32, pp. 19-42, 2003.

D. Shechtman, I. Blech, D. Gratias, and J.W. Cahn. Metallic Phase with Long-Range Orientational Order and No Translational Symmetry. *Physical Review Letters*, 53 pp. 1951-1953, 1984.

J.S. Shepherdson and H.E. Strurgis. Computability of Recursive Functions. *Journal of the Association of Computing Machinery*, 10, pp. 217-255, 1963.

J. R. Shoenfield. *Mathematical Logic*. Reading Mass : Addison-Wesley, 1967.

J. R. Shoenfield. *Recursion Theory*. Berlin : Springer-Verlag, 1993.

A. Shönhage. Storage Modification Machines. *SIAM Journal on Computing*, 9, (3), pp. 490-508, 1980.

P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithm on a Quantum Computer. *35th Annual Symposium on Foundations of Computer Science*, pp. 124-139, 1994.

W. Sieg. Church's Analysis of Effective Calculability. *The Bulletin of Symbolic Logic*, (2), Vol. 3 pp. 154-180, 1997.

W. Sieg. Calculations by Man and Machine : Mathematical Presentation. *Proceedings of the Cracow International Congress of Logic, Methodology and Philosophy of Science*, Synthese Series, Kluwer Academic Publishers, 2002.

W. Sieg and J. Byrnes. A Abstract Model for Parallel Computation : Gandy's Thesis. *Monist*, 82, pp. 150-164, 1999.

H.T. Siegelmann. Computation Beyond the Turing Limit. *Science*, 268 pp. 545-548, 1995.

H.T. Siegelmann. *Neural Networks and Analog Computation : Beyond the Turing Limit*. Boston : Birkhäuser, 1999.

M. Sipser. The History and Status of the P Versus NP Question. *24th Annual ACM*, pages 603–618, 1992.

P. Smith. *An Introduction to Gödel's Theorems*. Cambridge : Cambridge University Press, 2007.

W.D. Smith. Three Counterexamples Refuting Kieu's Plan for Quantum Adiabatic Computation ; and some Uncomputable Quantum Mechanical Tasks. *Applied Mathematics and Computation*, 178, pp. 184-193, 2006.

P. Smolensky. On the Proper Treatment of Connectionism. *Behavorial and Brain Sciences*, Vol. 11, pp. 1-23, 1988.

R. Solomonoff. A Formal Theory of Inductive Inference Part i. *Information and Control*, (1), 7 pp. 1-22, 1964.

M. Stannett. X-Machines and the Halting Problem : Building a Super-Turing Machine. *Formal Aspects of Computing*, 2 pp. 331-341, 1990.

M. Stannett. Hypercomputation is Experimentally Irrefutable. Technical report, Verication and Testing Research Group, Dept of Computer Science, University of Sheffield, 2001a.

M. Stannett. Computation over Arbitrary Models of Time. Technical report, Verication and Testing Research Group, Dept of Computer Science, University of Sheffield, 2001b.

M. Stannett. Computation and Hypercomputation. *Minds and Machines*, 13 pp. 115-153, 2003.

M. Stannett. Hypercomputational Models. In C. Teuscher, editor, *Alan Turing : Life and Legacy of a Great Thinker (2004)*. Berlin : Springer-Verlag, 2004.

M. Stannett. The Case of Hypercomputation. *Applied Mathematics and Computation*, 178, pp. 8-24, 2006.

M. Stannett. Specification, Testing and Verification of Unconventional Computations using Generalised X-Machines. *To appear in International Journal of General Systems*, 2013.

M. Stannett and I. Németi. Using Isabelle to Verify Special Relativity, with Application to Hypercomputation Theory. *arXiv :1211.6468*, 2012.

P.W. Stephens and A.I. Goldman. The Structure of Quasicrystals. *Scientific American*, April, pp. 44-53, 1991.

S. Stich. *From Folk Psychology to Cognitive Science*. Cambridge, MA : MIT Press, 1983.

K. Svozil. The Church-Turing thesis as a Guiding Principle for Physics. *Unconventional Models of Computation*, Singapore Springer-Verlag pp. 371-85, 1998.

A. Syropoulos. Some Thoughts on Hypercomputation. *Greek Molecular Computing Group*, ArXiv : 0910.1494v1, 2008.

L. Szilàrd. On the Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings. *Zeitschrift fur Physik*, 53 pp. 840-856, 1929.

J.F. Thomson. Tasks and Super-Tasks. *Analysis*, 15 pp. 1-10, 1954.

T. Toffoli. Cellular Automata as an Alternative to (rather than an Approximation of) Differential Equations in Modeling Physics. *Physica*, 10D, pp. 117-127, 1984.

A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1936.

A.M. Turing. Systems of Logic Based on the Ordinals. In M. Davis, editor, *The Undecidable (1965)*. Mineola, New York : Dover, 1939.

A.M. Turing. Proposal for Development in the Mathematics Division of an Automatic Computing Engine. In B.E. Carpenter and R.W. Doran, editors, *A.M. Turing's ACE Report of 1946 and Other Papers (1986)*. Cambridge, MA : MIT Press, 1945.

A.M. Turing. Computing Machinery and Intelligence. *Mind*, Vol. 59 (36), pp. 433-460, 1950.

A. Vaidya. The Epistemology of Modality. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2007.

J.P. Van Bendegem. Ross' Paradox is an Impossible Super-task. *British Journal for the Philosophy of Science*, 45 pp. 743-748, 1994.

J. Van der Spiegel, J.F. Tau, T. Alailima, and L.P. Ang. The ENIAC : History, Operation and Reconstruction in VSLI. In R. Rojas and U. Hashagen, editors, *The First Computers-History and Architectures*. Cambridge, MA : MIT Press, 2000.

S.B. Volchan. What is a Random Sequence. *The Mathematical Association of America*, 109 :46–63, 2009.

M. von Lambalgen. Von Mises' Definition of Random Sequences Reconsidered. *Journal of Symbolic Logic*, 52, pp. 725-755, 1987.

R. von Mises. On the Foundations of Probability and Statistics. *Annals of Mathematical Statistics*, 12, pp. 191-205, 1941.

R. von Mises. *Probability, Statistics and Truth*. New York : Dover, 1957.

J. von Neumann. First Draft of a Report on the EDVAC. Technical report, Philadelphia, PA : Moore School of Electrical Engineering, University of Pennsylvania, 1945.

J. von Neumann. Various Techniques Used in Connection with Random Digits. *Applied Mathematics Series*, 12, pp. 36-38, 1951.

P.L. Wantzel. Recherches sur les Moyens de Reconnaître si un Problème de Géométrie peut se Résoudre avec une Règle et un Compas. *Journal de Mathématiques Pures et Appliquées*, 2, pp. 366-372, 1837.

D.W. Watson. *Molecular Biology of the Gene*. Cold Spring Harbor Laboratory Press, 6 edition, 2007.

K. Weihrauch. *Computable Analysis : an Introduction*. Berlin : Springer-Verlag, 2000.

K. Weihrauch and N. Zhong. Is Wave Propagation Computable or can Wave Computers can Beat the Turing Machine ? *Proceedings of the London Mathematical Society*, 85 pp. 312-32, 2002.

P. Welch. On the Possibility, or Otherwise, of Hypercomputation. *The British Journal For The Philosophy of Science*, 55, pp. 739-746, 2004.

P. West. *Introduction to Supersymmetry and Supergravity*. Singapore : World Scientific, 1986.

H. Weyl. *Philosophie der* M*athematik und* N*aturwissenschaft*. Munich : R. Oldenbourg, 1927.

A.N. Whitehead and B.A. Russel. *Principia Mathematica*. Cambridge : Cambridge University Press, 2nd edition, 1927.

C. Whiteley. Minds, Machines and Gödel : a Reply to Mr. Lucas. *Philosophy*, 37 pp. 61-62, 1962.

M. Widom, K.J. Strandburg, and R.H. Swendsen. Quasicrystal Equilibrium State. *Physical Review Letters*, Vol. 58 (7), pp. 706-709, 1987.

B.R. Wilkins. *Analogue and Iterative Methods in Computation, Simulation, and Control*. London : Chapman and Hall, 1970.

L.J.J. Wittgenstein. Logisch-Philosophische Abhandlung. *Annalen der Naturphilosophische*, (3/4), 14, 1921.

G.K. Wolfgang. konrad Zuse's Plankalkül : The First High-Level 'non von Neumann' Programming Language. *IEEE Annals of the History of Computing*, 19, (2), pp. 17-24, 1997.

S. Wolfram. *A New Kind of Science*. Champaign, IL : Wolfram Media, 2002.

P. Wolper. *Introduction à la Calculabilité*. Paris : Dunod, 2006.

W.H. Woodin. *The Axiom of Determinacy, Forcing Axioms, and the Nonstationary Ideal*, volume 1 of *De Gruyter Series in Logic and Its Applications Logic and Its Applications Series*. W. de Gruyter, 1999.

R. Zach. Hilbert's Program. *Stanford Encyclopedia of Philosophy*, http ://plato.stanford.edu, 2003.

H. Zénil. Calcul et hyper-calcul. Master's thesis, Université Paris 1 Panth´éon-Sorbonne, 2006.

H. Zénil and F. Hernandez-Quiroz. On the Possible Computational Power of the Human Mind. In D. Aerts C. Gershenson and B. Edmonds, editors, *Philosophy and Complexity : Essays on Epistemology, Evolution and Emergence*. Singapor : World Scientific, 2006.

H. Zwirn. *Les Limites de la Connaissance*. Paris : Odile Jacob, 2000.

H. Zwirn and J.P. Delahaye. Unpredictability and computational irreducibility. In H. Zenil, editor, *Irreducibility and Computational Equivalence*, volume 2 of *Emergence, Complexity and Computation*, pages 273–295. Springer Berlin Heidelberg, 2013.