Calculabilité avancée
Thèse de Church-Turing

Kévin Perrot – Aix Marseille Université – 2024-25

Table des matières

1 Thèse de Church-Turing

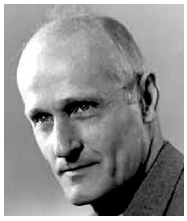
1.1 Un peu d'histoire


Cantor
(1845-1918)


Hilbert
(1862-1943)


Gödel
(1906-1978)


Church
(1903-1995)


Kleene
(1909-1994)


Turing
(1912-1954)


von Neumann
(1903-1957)

A la toute fin du XIX^{e} siècle, Georg Cantor définit les fondements de la théorie des ensembles, dont l'usage systématique (c'est-à-dire qui est utilisée dans tous les domaines) allait bouleverser les fondements de la logique mathématique. En 1900, pour fêter le passage au XX^{e} siècle, David Hilbert énonce 23 grands problèmes ouverts, dont le suivant : les propriétés qui s'expriment en langage mathématique sont-elles toutes décidables ? Si la réponse devait être affirmatives, les propriétés mathématiques valides seraient des théorèmes dérivables mécaniquement de quelques axiomes dans un système formel. Autrement dit : on pourrait remplacer les mathématicien·ne·s par des machines surpuissantes ! En 1931, Kurt Gödel met un terme à cette interrogation : il existe des propriétés mathématiques indécidables (dans tous les systèmes d'axiomes qui formalisent au moins l'arithmétique). Autrement dit : mathématicien·ne·s 1 - machines 0. Entre 1932 et 1936, Alsonso Church et Stephen Kleene proposent des modèles de calculs (le λ-calcul et les fonctions μ-récursives) qui semblent capturer la notion intuitive de fonctions calculables, mais il est un peu difficile de s'en convaincre... Notons tout de même que le λ-calcul est extrêmement minimaliste, ce qui rend sa compréhension mathématique fort intéressante :

tout est capturé en quelques lignes de définition ! Indépendamment, en 1936, Alan Turing propose sa définition de machines. En 1937 il montre que la classe des fonctions λ-calculables est égale à la classe des fonctions programmables sur les machines de Turing. Les machines de Turing permettent de reformuler en termes intuitifs de calculs les résultats de Kurt Gödel (qui étaient exprimés en termes de démonstration). Avec l'aide de Von Neumann (et d'autres), les premiers ordinateurs programmables verront le jour quelques années plus tard !
La vie de Turing vaut le coup d'oeil (savez vous que le rôle de Turing durant la seconde guerre mondiale est resté secret d'Etat de nombreuses années ?).
e-penser (13') : `https://www.youtube.com/watch?v=7dpFeXV_hqs`

1.2 Thèse de Church-Turing

Nous avons vu que les machines de Turing ne peuvent pas calculer toutes les fonctions, et même qu'elles ne sont capables d'en calculer qu'une infime partie. Il est légitime de se poser les questions suivantes : est-ce un bon modèle de calcul ? Ne pourrions nous pas définir un modèle de calcul qui puisse calculer un plus grand ensemble de fonctions ?

Définition 1. *Deux modèles de calcul sont* **équivalents** *s'ils sont capables de se simuler mutuellement (donc ils calculent exactement le même ensemble de fonctions).*

Remarque 2. *Les MT non déterministes et multi-rubans sont équivalentes aux MT.*

Il se trouve que tous les modèles de calcul « réalistes » qui ont été définis jusqu'à aujourd'hui sont équivalents aux machines de Turing : ils permettent de calculer exactement le même ensemble de fonctions. *Exactement* le même ensemble de fonctions !

Historiquement, en 1933 Kurt Gödel et Jacques Herbrand définissent le modèle des fonctions μ-récursives. En 1936, Alonzo Church définit le λ-calcul. En 1936 (sans avoir connaissance des travaux de Church), Alan Turing propose sa définition de machine. Church et Turing démontrent alors que ces trois modèles de calcul sont équivalents !

La thèse de Church-Turing, ou plutôt ses deux versions [3], sont des énoncés que la communauté (dans sa majorité) pense vrais, mais qu'il n'est pas possible (du moins c'est le point de vue jusqu'à maintenant) de prouver. Ils énoncent que les machines de Turing capturent « correctement » la notion de calcul : toute autre façon de calculer, ou de définir le calcul, reviendrait au même [1].

1.3 Thèse de Church-Turing version physique

Thèse 3. *Toute fonction physiquement calculable est calculable par une MT.*

Autrement dit tout modèle de machine, qui peut effectivement exister selon les lois de la physique, sera au mieux équivalent aux machines de Turing, sinon moins puissant (en terme d'ensemble de fonctions calculables).

Robin Gandy (qui a effectué son doctorat sous la direction d'Alan Turing) a travaillé sur cette question et démontré un résultat que nous reproduisons ci-dessous dans une formulation simplifiée [2].

1. les machines quantiques n'échappent pas à la thèse de Church-Turing [1].

Théorème 4. *Toute fonction calculée par une machine respectant les lois physiques :*

1. *homogénéité de l'espace (partout les mêmes lois),*
2. *homogénéité du temps (toujours les mêmes lois),*
3. *densité d'information bornée (pas plus de n bits au m^2),*
4. *vitesse de propagation de l'information bornée (pas plus de c $m.s^{-1}$),*
5. *quiescence (configuration initiale finie et état de repos tout autour),*

est calculable par une machine de Turing.

Est-ce satisfaisant ? Une version quantique de ce théorème a été démontrée [1].

1.4 Thèse de Church-Turing version symbolique

Thèse 5. *Toute fonction calculée par un algorithme est calculable par une MT.*

Pas facile de définir ce qu'est, ou plutôt ce qu'en toute généralité pourrait être, un **algorithme**. On peut dire qu'un algorithme est exprimable par un programme rédigé dans un certain langage de programmation, qu'il décrit des instructions pouvant être suivies sans faire appel à une quelconque « réflexion ». Définir un modèle de calcul revient à définir une façon d'écrire des algorithmes.

Cette version de la thèse de Church-Turing est parfois appelée version symbolique, car elle exprime ce qu'il est possible de calculer à l'aide de symboles mathématiques auxquels on donne un sens calculatoire.

Rappelons qu'Alan Turing est parti de l'idée d'un calculateur humain avec son stylo devant une feuille de papier, pour construire le modèle des machines qui portent son nom. Les machines de Turing sont-elles assez générales ? ou bien peut-on imaginer une autre façon non-équivalente de décrire les algorithmes, mais qui soit en accord avec notre intuition ? La Thèse 5 postule que les machines de Turing capturent en toute généralité la notion d'algorithme, et que l'on peut s'en contenter.

Malgré de nombreux efforts, la thèse de Church-Turing n'a jusqu'ici jamais été contredite !

`https://fr.wikipedia.org/wiki/Th%C3%A8se_de_Church`

1.5 Lancer des machines de Turing dans des trous noirs ?

Pas facile : `https://interstices.info/calculer-differemment/`

1.6 Turing-complétude

On appelle **modèle de calcul** la définition mathématique d'un ensemble d'opérations utilisables pour réaliser un calcul (une syntaxe et des règles décrivant la sémantique de la syntaxe). Exemple : les machines de Turing.

Définition 6. *Un modèle de calcul capable de calculer toutes les fonctions calculables par des machines de Turing est appelé* **Turing-complet**.

Remarque 7. *Un modèle de calcul capable de simuler une machine de Turing universelle est Turing-complet.*

Tous les langages de programmation que vous utilisez couramment (C, Haskell, Java, OCaml, Python...) sont bien entendu Turing-complets : si vous pouvez implémenter un simulateur de machines de Turing, alors le langages est capable de calculer toutes les fonctions calculables par des machine de Turing !

Petite liste de modèles, jeux et langages Turing-complets (parfois accidentellement) : `https://en.wikipedia.org/wiki/Turing_completeness#Unintentional_Turing_completeness`

- le λ-calcul (très minimaliste),
- les fonctions μ-récursives (façon calculatoire de définir des fonctions),
- les pavages (un type de puzzles),
- les automates cellulaires (vit-on dans un AC ?),
- les jeux Minecraft et Pokemon jaune (et de nombreux autres),
- Brainf*ck (un langage extrêmement simpliste qui comporte 8 instructions).
- Les briques de Lego™mécaniques (avec engrenages et pistons) : `http://www.dailymotion.com/video/xrmfie/`.

Références

[1] P. Arrighi and G. Dowek. The physical Church-Turing thesis and the principles of quantum theory. *Int. J. of Found. of C.S.*, 23 :1131–1145, 2012. arXiv:1102.1612.

[2] R. Gandy. Church's Thesis and Principles for Mechanisms. *The Kleene Symposium*, 101 :123–148, 1980.

[3] M. Pégny. Les deux formes de la thèse de Church-Turing et l'épistémologie du calcul. *Philosophia Scientiae*, 16(3) :39–67, 2012.