Peut-on remplacer le raisonnement humain par des algorithmes ?

Christine Solnon

▶ **To cite this version:**

Christine Solnon. Peut-on remplacer le raisonnement humain par des algorithmes ?. La Recherche, 2025, 580. hal-04881777

HAL Id: hal-04881777
https://hal.science/hal-04881777v1

Submitted on 12 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Peut-on remplacer le raisonnement humain par des algorithmes ?

Christine Solnon, CITI, Inria, INSA Lyon, F-69621 Villeurbanne

Chronique parue dans La recherche N° 580, janvier-mards 2025

Les progrès récents des intelligences artificielles (IA), notamment l'IA générative (qui produit du texte ou des images en réponse à un prompt), sont impressionnants. Ils font reculer les limites de l'imaginable, à tel point que l'on entend de plus en plus souvent parler d'IA générales, capables de résoudre des problèmes de natures très variées. Alors, peut-on complètement automatiser le raisonnement, ou bien s'agit-il de science-fiction ? Qu'en disent les mathématiques et la logique ? Les premiers éléments de réponse remontent à l'Antiquité grecque. C'est alors que la logique est introduite pour faciliter le raisonnement et l'élaboration de discours cohérents. Ainsi, les syllogismes d'Aristote permettent de dériver des conclusions à partir de prémisses : si tous les hommes sont mortels, et si Socrate est un homme, alors nous pouvons en déduire que Socrate est mortel. Ce genre de raisonnement logique est également utilisé en mathématiques pour démontrer qu'un théorème est une conséquence d'un ensemble donné d'axiomes. Par exemple, dans les Éléments (écrit au IVe siècle avant notre ère), Euclide définit sa géométrie en se fondant sur cinq axiomes (définissant les notions de segment de droite, de droite, de cercle, d'angle droit et de parallélisme). C'est à partir de ces axiomes que l'on peut démontrer des théorèmes, comme celui de Pythagore (dans un triangle rectangle, le carré de la longueur du plus grand côté est la somme des carrés des deux autres côtés), ou celui de Thalès (dans un triangle, une droite parallèle à l'un des côtés définit avec les droites des deux autres côtés un nouveau triangle, dont les longueurs sont proportionnelles à celles du premier triangle).

Une étape majeure est franchie au XIXe siècle, lorsque la logique est définie en termes mathématiques. En particulier, les travaux du mathématicien et philosophe irlandais George Boole (1815-1864) rendent possible le remplacement de certains raisonnements par des calculs portant sur des variables logiques, « vraies » ou « fausses » - un peu comme on fait des opérations

arithmétiques sur des variables numériques. Cette formalisation de la logique ouvre la porte vers l'automatisation du raisonnement. Puis, au début du XXe siècle, le mathématicien allemand David Hilbert (1862-1943) défie la communauté scientifique de trouver un système formel axiomatique, qui soit à la fois cohérent (ne permettant pas de démontrer deux théorèmes contradictoires) et complet (permettant de démontrer tous les théorèmes mathématiques). Prenons un exemple en nous restreignant à la petite partie des mathématiques, qui concerne l'addition de deux entiers naturels. Dans ce cadre, nous pouvons commencer par définir récursivement l'ensemble $\mathbb{N}$ des entiers naturels de la façon suivante : 0 appartient à $\mathbb{N}$, et si x appartient à $\mathbb{N}$, alors $s(x)$ appartient aussi à $\mathbb{N}$ ($s(x)$ s'interprétant en « le nombre suivant x »). On peut ensuite introduire deux axiomes pour définir l'addition : (a) pour tout x appartenant à $\mathbb{N}$, $0 + x = x$, et (b) pour tout x, y, et z appartenant à $\mathbb{N}$, si $x + y = z$, alors $s(x) + y = s(z)$. Avec un tel système, nous prouvons que l'assertion « $s(s(s(0))) + s(s(0)) = s(s(s(s(s(0)))))$ » est vraie (autrement dit, « $3 + 2 = 5$ » est un théorème), en montrant qu'elle est une conséquence logique des axiomes : on part de l'axiome (a) en remplaçant x par $s(s(0))$, puis on applique deux fois l'axiome (b) pour arriver jusqu'à notre assertion.

Ce petit système est une version simplifiée de celui introduit en 1929 par le mathématicien polonais Mojzesz Presburger (1904-1943). Cela peut sembler anecdotique, mais il s'agit en fait d'une révolution car il devient alors possible de prouver de façon automatique des théorèmes. En d'autres termes, on peut remplacer le raisonnement humain par des calculs (1). Tous les raisonnements ? Non, car le mathématicien et logicien autrichien Kurt Gödel (1906-1978) douche cet espoir en démontrant en 1931 un célèbre théorème : un système formel cohérent pour l'arithmétique des nombres entiers ne peut pas être complet. Autrement dit, il n'est pas possible de démontrer de façon automatique tous les théorèmes de l'arithmétique des nombres entiers, car certaines assertions ne peuvent être ni confirmées ni infirmées en partant des axiomes. C'est ce qu'on appelle des assertions « indécidables ». Se pose alors la question d'identifier le périmètre de ce qui est effectivement calculable. Par calculable, on entend qu'il existe un algorithme qui résoudrait le problème de façon automatique. Deux réponses sont apportées la même année, en 1936 : le mathématicien britannique Alan Turing (1912-1954) introduit une machine théorique, tandis que le mathématicien états-unien Alonzo Church (1903-1995) introduit le lambda-calcul.

Ces deux formalismes sont apparemment très différents : la machine de Turing est un automate qui enchaîne des transitions entre états, conditionnées par un symbole lu sur un ruban, et est capable d'écrire de nou-

veaux symboles sur ce ruban, tandis que le lambda-calcul est un langage de programmation théorique qui permet de définir des fonctions et d'appliquer ces fonctions à des arguments, un peu comme lorsqu'on appelle une fonction dans un langage de programmation classique (les langages tels que Lisp ou Caml sont les héritiers directs du lambda-calcul). Church et Turing ont démontré l'équivalence de ces deux formalismes : ce qui est calculable par l'un l'est aussi par l'autre. A-t-on atteint à cette époque, avant même que les premiers ordinateurs ne soient conçus, un concept absolu de calcul indépendant d'un formalisme ? Le mathématicien britannique Robin Gandy (1919-1995) le démontre en 1980 : ce qui ne peut pas être calculé par la machine théorique de Turing ne peut être calculé par aucune machine physique (y compris des ordinateurs quantiques). Ces problèmes sont indécidables et il en existe beaucoup, dont certains sont fondamentaux, comme la détection automatique des bugs d'un programme.

Nous pouvons utiliser une IA pour résoudre un problème indécidable, mais les travaux de Gödel, Church et Turing nous disent que les réponses ne peuvent pas être à la fois correctes et complètes. C'est-à-dire qu'il existera nécessairement des cas où l'IA répondra quelque chose d'erroné, ou bien ne saura pas répondre. Si les IA génératives actuelles ont tendance à donner des réponses fausses (les « hallucinations »), il existe des approches qui sont capables de garantir la correction au prix de la complétude. Ainsi, le domaine de l'informatique que l'on appelle l'interprétation abstraite, initié à la fin des années 1970 par les informaticiens français Patrick et Radia Cousot, est une approche incomplète qui est utilisée pour prouver que les logiciels embarqués dans certains avions ne contiennent pas de bugs (entre autres) : quand l'approche conclut qu'il n'y a pas de bug, nous pouvons être certains que c'est bien le cas ; dans d'autres cas, elle lance des « alarmes » pouvant correspondre soit à des bugs soit à des fausses alertes, et c'est au programmeur de modifier son code jusqu'à ce qu'il n'y ait plus d'alarmes. Les programmes de vols d'avion, du spatial, et autres logiciels dits « temps réel critique » passent systématiquement par cette approche.

Pour finir, revenons en 1931 et à Gödel. Son théorème d'incomplétude comporte un corollaire intéressant à rappeler : un système formel cohérent ne peut pas démontrer sa propre cohérence. Par conséquent, si une IA était cohérente (dans le sens où elle ne se contredirait jamais), alors ses capacités d'introspection - et donc ses facultés de conscience - seraient nécessairement limitées. Rassurant ?

[1] Gilles Dowek, Les Métamorphoses du calcul, Le Pommier, 2007.