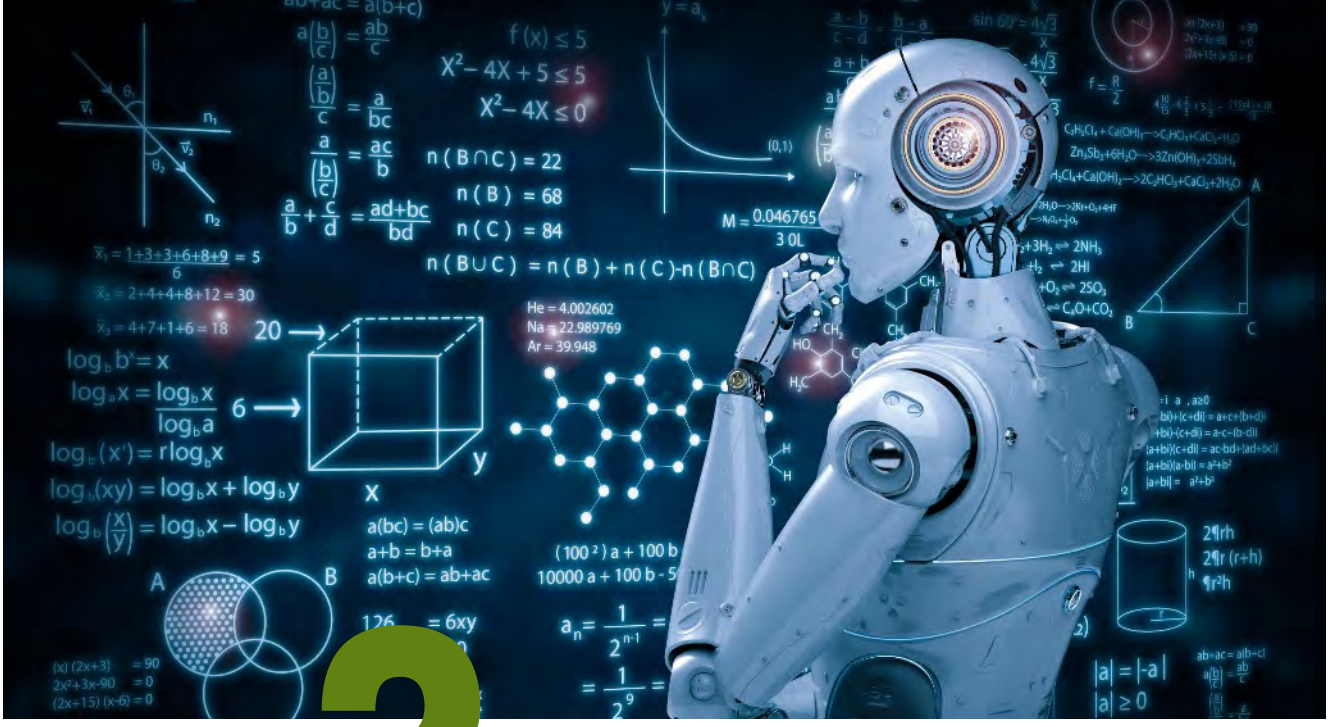

2

l'infini des vérités faciles et trop proches les unes des autres. L'intelligence artificielle peut-elle concevoir des méthodes qui égalent les mathématiciens dans le travail d'imagination et de construction ordonnée des théories mathématiques, qui ne sont pas des successions ennuyeuses de vérités triviales comme 1+1=2, 1+2=3, 1+3=4, etc.?

Nous allons voir que la «recherche automatisée» progresse petit à petit: il se peut que, imperceptiblement, nous entrions dans une nouvelle ère.

PREMIÈRES TENTATIVES

Le logicien Hao Wang, un ami d'Einstein à Princeton, écrivit en 1959 un programme pour produire et sélectionner des théorèmes du calcul propositionnel, une partie de la logique où l'on traite de formules du type (A et B)⇒A, (A ou non-A), (A⇒B)⇒(non-B⇒non-A). Cette tentative lui fit prendre conscience que le problème principal était de sélectionner les résultats «intéressants», mais le domaine trop étroit du calcul propositionnel fit qu'aucun résultat spectaculaire ne sortit de cette première tentative.

En juillet 1976, Douglas Lenat soutint à Stanford un mémoire de doctorat dont le sujet était son «mathématicien automatique», le programme AM (pour Automated Mathematician). Il y décrivait un programme en langage LISP, qui, à partir de règles et de manipulations symboliques, déduit et accumule des concepts et des propriétés de ces concepts. Douglas Lenat explique que son programme a découvert la notion de nombre premier, a énoncé le théorème de décomposition d'un nombre en facteurs premiers. Il aurait aussi redécouvert la fameuse conjecture >

FAIRE DES CONJECTURES INTÉRESSANTES

Pour concevoir des systèmes menant une recherche mathématique, il faut faire en sorte que ces systèmes effectuent des tris. Un tel système ne doit retenir que les propriétés et conjectures les plus intéressantes.
Le Britannique Simon Colton, qui a développé le système HR *(voir le texte principal)*, propose six critères pour repérer les conjectures intéressantes.

1) Les conjectures doivent être plausibles. Il faut éliminer celles qui sont fausses parce qu'il existe des exemples simples qui les contredisent.

2) Les conjectures doivent être nouvelles. Le système doit disposer des résultats connus sur le sujet étudié et savoir les exploiter pour ne pas proposer les conséquences immédiates de résultats connus.

3) Les résultats envisagés doivent être étonnants. John Conway pense qu'un résultat est d'autant plus intéressant qu'« il est scandaleux » *(outrageous)*. Il est délicat de concevoir des programmes sachant définir ce qui est « étonnant ». Le plus souvent, on se contente d'éliminer les énoncés qui sont des conséquences immédiates d'énoncés connus ou qui sont des cas particuliers de formules logiques toujours vraies, comme non(non-A) ⇔ A. Une propriété simple qui exige une longue démonstration est étonnante puisqu'on ne peut pas comprendre aisément ce qui la rend vraie. Malheureusement, la notion : « exige une longue démonstration » est impossible à programmer en pratique, car, pour l'affirmer d'un énoncé, il faudrait explorer tous les chemins de démonstrations possibles pouvant y conduire.

4) Les conjectures doivent concerner de nombreux cas. Si l'on s'intéresse à des propriétés P_1 et P_2, la conjecture $P_1 \Rightarrow P_2$ n'est intéressante que si une grande proportion des objets jouit de la propriété P_1.

5) Les conjectures doivent être facilement compréhensibles et donc doivent avoir des énoncés assez courts et si possible ne faire intervenir que les concepts de base de la théorie étudiée.

6) Les conjectures doivent être fécondes et par exemple avoir beaucoup de conséquences simples. En mathématiques, il est difficile d'anticiper cette qualité d'une conjecture. Le grand théorème de Fermat n'est considéré comme fécond que parce qu'il a résisté pendant des siècles. Si Fermat l'avait démontré en quelques lignes, personne n'aurait estimé l'énoncé de ce théorème comme intéressant.

Sur ce sujet, voir :
S. Colton *et al.*, « On the notion of interestingness in automated mathematical discovery », *International Journal of Human-Computer Studies*, vol. 53(3), pp. 351-375, 2000.

de Goldbach, toujours non démontrée («Tout nombre pair à partir de 4 est somme de deux nombres premiers»). Cependant, ces affirmations ont été discutées car elles résultent d'une interprétation indirecte *a posteriori* des calculs d'AM et parce qu'il est difficile de savoir si ce qui est présent dans le programme avant sa mise en route n'est pas déjà suffisamment déterminant pour que de telles «découvertes» se produisent nécessairement.

Ce problème de savoir si un système fait vraiment une découverte ou s'il est forcé de la faire par les mécanismes simples programmés au départ se retrouvera pour tous les systèmes qui ont succédé à cette tentative de mathématicien automatique. La résolution de cette difficulté ne peut provenir que de la découverte par la machine d'idées ou de propriétés nouvelles et reconnues intéressantes par les mathématiciens eux-mêmes.

Après ce premier essai, Douglas Lenat travailla sur un système nommé Eurisko, lui aussi destiné à produire des découvertes mathématiques. Son originalité est qu'il créait spontanément des règles heuristiques nouvelles à l'aide de métarègles du type: «Donner la priorité aux règles les plus simples». Face aux difficultés à obtenir des résultats concluants, Douglas Lenat entreprit de concevoir et programmer un système dénommé Cyc, qui maîtrise des connaissances générales de tout humain, ce qu'il considérait comme indispensable pour aller plus loin en intelligence artificielle (IA). Pour lui, un système intelligent doit savoir que «si un humain marche, il ne reste pas à la même place» ou que «quand il pleut, les passants dans la rue sont mouillés et certains ouvrent des parapluies», etc.

Le projet Cyc, démarré en 1983, a bénéficié depuis d'un travail équivalent à plus de mille hommes-années et se poursuit aujourd'hui encore. C'est le plus ambitieux et long projet d'intelligence artificielle jamais mené. Il a développé un savoir général et plusieurs savoirs particuliers liés à des sous-domaines, dont celui du terrorisme qui est probablement utilisé par l'agence américaine NSA (National Security Agency). Le système connaît plus d'un million de termes comme «la France», «le kilogramme», «New York», «les poissons», «la chasse»... et plus de 24 millions d'affirmations de base les concernant et les liant. Elles sont écrites dans un langage logique et disposent d'un système de déduction qui permet, à partir des affirmations de base, d'en produire de nouvelles.

Cette recherche appartient à l'«IA symbolique»: les méthodes ne s'appuient pas sur des réseaux de neurones, ce sont des programmes manipulant des symboles, des mots, des concepts représentés par des éléments de programmes. Les humains qui les écrivent comprennent le détail de ce qu'ils y mettent et de ce qui se produit quand la machine fonctionne, contrairement aux réseaux de neurones de l'«apprentissage profond», où le système n'explique pas ses réponses. Le lien avec les mathématiques est devenu secondaire dans le projet Cyc, bien qu'un module ait été conçu pour aider les étudiants en mathématiques.

Depuis les succès des approches de type apprentissage profond, par exemple pour le jeu de go, le projet Cyc est regardé avec méfiance. Pourtant, il semble certain que les progrès de l'IA devront associer la capacité de raisonnement et les connaissances générales de systèmes comme Cyc avec des méthodes de type statistique et neuronal.

RECHERCHE DE CONCEPTS

Quelques années après Eurisko, Simon Colton, de l'université d'Edimbourg, a réussi à faire «imaginer» des notions mathématiques nouvelles à son programme, dénommé HR en l'honneur des mathématiciens Godfrey Hardy et Srinivasa Ramanujan, génie ayant découvert

3 LES NOMBRES REFACTORISABLES

Parmi plusieurs idées arithmétiques analogues, celle des nombres « refactorisables » a été proposée par le programme HR de Simon Colton. Un nombre entier n est refactorisable s'il est divisible par le nombre de ses diviseurs. Si p est un nombre premier, alors p^{p-1} est refactorisable, car ses diviseurs sont 1, p, p^2, p^3, ..., p^{p-1}, qui sont au nombre de p. Cette propriété arithmétique avait déjà été envisagée indépendamment par d'autres chercheurs, ce qui confirme que c'est une idée mathématique intéressante. Les premiers nombres refactorisables sont : 1, 2, 8, 9, 12, 18, 24, 36, 40, 56, 60, 72, 80, 84, 88, 96, 104, 108, 128, 132, 136, 152, 156, 180, 184, 204, 225, 228, 232, 240, 248, 252, 276, 288, 296, 328, 344, 348, 360, 372...

Voici cinq théorèmes sur les nombres refactorisables, dont les trois derniers ont été découverts par HR :

Théorème 1. Il existe une infinité de nombres refactorisables pairs, et une infinité de nombres refactorisables impairs.

Théorème 2. Tous les nombres impairs refactorisables sont des carrés.

Théorème 3. Un nombre parfait (égal à la somme de ses diviseurs propres, comme $28 = 1 + 2 + 4 + 7 + 14$) n'est jamais refactorisable.

Théorème 4. Si n est refactorisable, alors n est de la forme $8k$, $8k + 1$, $8k + 2$ ou $8k + 4$.

Théorème 5. La somme des diviseurs d'un nombre impair refactorisable est elle-même impaire.