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

LES NOMBRES REFACTORISABLES

3

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 8*k*, 8*k* + 1, 8*k* + 2 ou 8*k* + 4.

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

une multitude de formules extraordinaires qui n'ont parfois été démontrées qu'après sa mort.

Le programme HR s'est intéressé à l'arithmétique et a tenté à la fois de créer des concepts nouveaux – des définitions de suites numériques nouvelles –, de formuler des hypothèses à leur sujet et parfois de les démontrer à l'aide de programmes de démonstration automatique qu'il mettait à son service. Le programme disposait de critères pour ne se concentrer que sur des définitions potentiellement intéressantes. Il a pu ainsi identifier 14 suites numériques nouvelles, qui ont été acceptées en 2000 par l'Encyclopédie des suites numériques de Neil Sloane (http://oeis.org), où ne figurent que les suites ayant de l'intérêt aux yeux des mathématiciens.

L'une des suites proposées par HR est étonnamment simple et a conduit Simon Colton à publier un article dans le *Journal of Integer Sequences* en 1999. Il s'agit de la suite des «nombres refactorisables». Un entier n est refactorisable s'il est divisible par le nombre de ses diviseurs. L'entier 9 est par exemple refactorisable, car il a 3 diviseurs (1, 3 et 9) et que justement il est divisible par 3. Plus généralement, si p est premier, p^{p-1} est refactorisable.

À vrai dire, cette invention particulière du programme de Colton n'en n'était pas tout à fait une, car après la publication de l'article de 1999, on s'est aperçu qu'un autre article publié en 1990 avait envisagé la même notion… HR n'était donc pas le premier à s'intéresser à la notion, mais cela confirme que l'idée proposée est réellement jugée intéressante par les mathématiciens.

Notons que le programme de Simon Colton a découvert tout seul cette jolie propriété: «Si la somme des diviseurs d'un entier est un nombre premier, alors le nombre de diviseurs de ce nombre est aussi un nombre premier.»

Le programme HR a aussi inventé des notions concernant les groupes, les anneaux finis et les graphes. Il dispose de diverses mesures évaluant l'intérêt d'un concept, se concentre sur le plus prometteur et cherche alors à énoncer des propriétés le concernant pour lesquelles il tente de trouver des démonstrations ou des contre-exemples. Cette façon de procéder est proche de ce que font souvent les chercheurs en mathématiques, mais partiellement car un chercheur prend de la distance vis-à-vis de son sujet et tente d'établir des liens entre concepts appartenant à des champs éloignés.

UN VRAI SUCCÈS SUR LES GRAPHES

Le principe général de HR est particulièrement efficace quand il s'agit de structures finies, car le programme peut disposer d'une liste variée d'exemples et les utiliser pour tester si les propriétés qu'il envisage sont vérifiées par chaque élément de sa base d'exemples. Si c'est le cas, la propriété devient plausible et le système l'envisage comme conjecture. Si l'un des exemples >

4 LA MÉTHODE GRAFFITI

Le programme Graffiti énumère et sélectionne des conjectures portant sur les invariants des graphes.

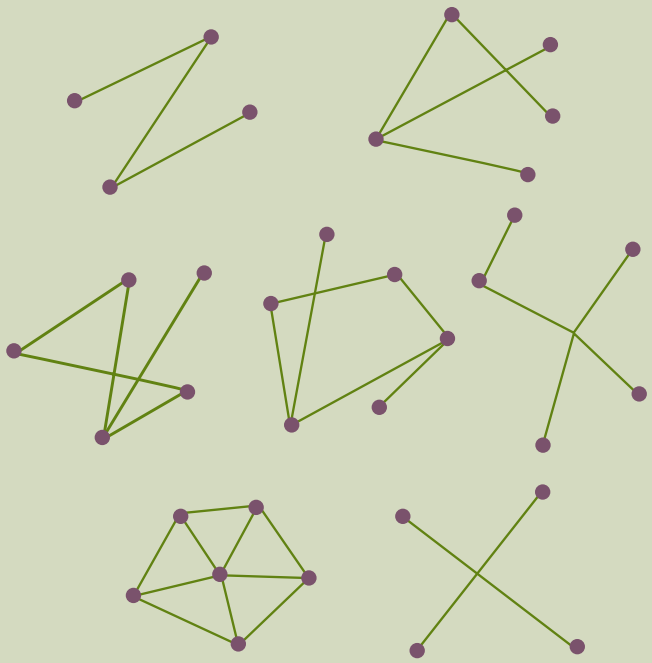
- Le programme considère les invariants inv_1, inv_2, …, inv_k et dispose pour chacun d'un sous-programme qui le calcule. Parmi les invariants possibles, il y a le nombre de nœuds du graphe, le nombre maximal de liens d'un nœud du graphe avec les autres nœuds, etc.

- Le programme dispose d'une base de test formée des graphes G_1, G_2, …, G_h. Cette base peut être complétée au fur et à mesure de l'utilisation du système pour le rendre plus efficace.

- Le programme dispose d'un système qui énumère les relations algébriques entre invariants. Ce système engendre par exemple $inv_1 + inv_2 \leq inv_3$, $inv_5/inv_4 > (inv_1)^2$, etc. Le système donnera la priorité aux relations simples.

- La recherche de conjectures portant sur les invariants consiste alors à tester les relations algébriques énumérées par le système. Celui-ci élimine toute relation non vérifiée par l'un des graphes de la base de test.

- Les relations algébriques passant les tests sont soumises à une seconde série de tests pour déterminer si elles sont connues ou résultent de relations connues. Les relations non écartées sont proposées comme conjectures aux mathématiciens, qui tentent de les démontrer.



Quelques exemples de graphes. Le système Graffiti s'appuie sur une base constituée de nombreux graphes pour tester des relations algébriques entre les «invariants» (nombre de nœuds d'un graphe, nombre moyen d'arêtes entre deux nœuds, etc.).