

se faire une idée, voici quelques détails sur le problème et sur le travail de l'ordinateur de W. McCune.

LES ALGÈBRES DE BOOLE

La conjecture de Robbins indique que toute structure définie par une opération d'addition, $x+y$, et une opération de complémentation, $n(x)$, qui vérifie les trois propriétés suivantes :

- commutativité (C) : $x+y = y+x$
- associativité (A) : $(x+y)+z = x+(y+z)$
- équation de Robbins (R) :

$$n(n(x+y) + n(x+n(y))) = x,$$

vérifie nécessairement toutes les propriétés que possède l'opération de réunion ensembliste et celle de complémentation dans un ensemble.

Les structures vérifiant les axiomes (C), (A) et (R) s'appellent des *algèbres de Robbins*. Celles vérifiant les propriétés que possèdent la réunion et la complémentation dans un ensemble

(liste indiquée sur la figure 3) s'appellent des *algèbres de Boole*, du nom de George Boole, mathématicien irlandais du XIX^e siècle, auteur du célèbre ouvrage *Les lois de la pensée*, qui décrit un calcul algébrique nouveau à la base de la logique moderne et de la théorie des ensembles.

La conjecture de Robbins affirme donc que «toute algèbre de Robbins est une algèbre de Boole», et la démontrer consiste à utiliser uniquement les trois propriétés (C), (A) et (R) pour en déduire celles de la figure 3. Essayez, par exemple, de déduire $x + x = x$. Il est facile de montrer qu'une algèbre de Boole est une algèbre de Robbins ; c'est la réciproque qui est difficile.

En fait, l'ordinateur n'a pas démontré exactement cette réciproque. Il a utilisé un résultat établi par S. Winker au début des années 1980 : si, dans une algèbre de Robbins, on peut trouver deux éléments a et b vérifiant l'équa-

tion de Winker (W) : $n(a + b) = n(a)$, alors l'algèbre en question est une algèbre de Boole.

L'ordinateur n'a donc eu qu'à établir l'implication : si (C), (A) et (R), alors il existe a et b tels que (W).

Bien des tentatives avaient été faites à la main, mais personne n'avait jamais réussi. Le programme EQP (*Equational Prover*) de W. McCune aboutit en octobre 1996, après un calcul ininterrompu d'une durée de huit jours sur une machine utilisant un processeur RS/6000, en consommant 30 mégaoctets de mémoire et en passant en revue plus de 2 612 977 expressions intermédiaires.

L'assaut final fut précédé de plusieurs autres, durant lesquels le programme était paramétré de façon différente et dont l'examen des résultats a conduit à un ajustement judicieux. Si on prend en compte la durée de cette phase de mise au point de la recherche, le déroulement du calcul a occupé le puissant microprocesseur cinq semaines entières.

4. LA DÉMONSTRATION DU 2 OCTOBRE 1996

Voici la preuve de la conjecture de Robbins trouvée par programme. La première étape a été détaillée. Pour les autres, des indications sommaires sont données. La dernière formule en changeant l'ordre de quelques termes est de la forme $n(a + b) = n(a)$; elle montre donc qu'il existe a et b vérifiant l'équation de Winker. Les numérotations des formules sont celles du programme. Les étapes inutiles ont été omises, d'où la discontinuité de la numérotation.

3 [équation de Robbins] $n(n(n(x)+y)+n(x+y)) = y$.

5 [3,3] $n(n(n(x+y)+n(x+y)+y) = n(x+y)$.

Le crochet [3,3] indique qu'on est parti de l'équation 3 et qu'on a simplifié grâce à 3. Voici le détail de cette première étape :

- on part de 3 : $n(n(n(x)+y)+n(x+y)) = y$.

- on remplace x par $n(x)+y$ et y par $n(x+y)$; cela donne :

$$n(n(n(n(x)+y)+n(x+y)+n(x+y)+y)) = n(x+y)$$

- on reconnaît le premier membre de l'équation de Robbins, qu'on

remplace par y : $n(y+n(n(x)+y)+n(x+y)) = n(x+y)$

- on change l'ordre ce qui est autorisé à cause de l'axiome (C),
 $n(n(n(x+y)+n(x+y)+y) = n(x+y)$

6 [3,3] $n(n(n(n(x)+y)+x+y)+y) = n(n(x)+y)$.

24 [6,3] $n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y))) = y$.

47 [24,3] $n(n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+z)+n(y+z)) = z$.

48 [24,3] $n(n(n(n(n(x)+y)+n(n(x)+y)+x+y+y)+y) = n(n(x)+y)$.

146 [48,3] $n(n(n(n(n(x)+y)+n(n(x)+y)+x+y+y+y)+n(n(x)+y))) = y$.

250 [47,3] $n(n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z) = n(y+z)$.

996 [250,3] $n(n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)$

$$+ n(y+z)+z)+z+u)+n(n(y+z)+u)) = u.$$

16379 [5,996,3] $n(n(n(n(x)+x)+x+x+x)+x) = n(n(x)+x)$.

16387 [16379,3] $n(n(n(n(n(n(x)+x)+x+x+x)+x+y)+n(n(n(x)+x)+y))) = y$.

16388 [16379,3] $n(n(n(n(n(x)+x)+x+x+x+x)+n(n(x)+x))) = x$.

16393 [16388,3] $n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x) = n(n(x)+x)$.

16426 [16393,3] $n(n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+y)$
 $+ n(n(n(x)+x)+y))) = y$.

17547 [146,16387] $n(n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+n(n(n(x)+x)$
 $+x+x+x)+x)+x) = n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)$.

17666 [24,16426,17547] $n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)$
 $= n(n(n(x)+x)+x+x+x+x)$.

VÉRIFICATION ET MISE À DISPOSITION DES DONNÉES

Les deux millions de termes manipulés pendant l'assaut d'octobre ne sont pas tous indispensables à l'écriture de la démonstration que (C), (A) et (R) implique (W). Loin s'en faut ! Ce que l'ordinateur a trouvé se résume en quatre pages vérifiables à la main, contrairement aux deux exemples de démonstration sur ordinateur cités précédemment. Une esquisse de cette démonstration est donnée en figure 4.

Le programme vainqueur EQP est un programme de démonstration automatique spécialisé qui a été développé à partir d'un autre programme de W. McCune, plus général, appelé OTTER. Pour passer du premier programme à EQP, on a amélioré l'efficacité du traitement des équations dans une algèbre où l'opération de base est commutative et associative, et l'on a multiplié les «paramètres de réglages».

Ces paramètres déterminent la stratégie de manipulation des équations que le programme engendre et combine lors de sa recherche. L'un d'eux, par exemple, indique qu'il ne faut pas prendre en considération les équations de longueur supérieure à une borne fixée. Ce paramètre était fixé à 70 dans la recherche d'octobre 1996. Quand ce paramètre est fixé trop bas, la recherche n'aboutit pas, car la démonstration recherchée passe par de longues équations ; quand le paramètre est fixé trop

haut, la recherche s'égare et n'aboutit pas, non plus, dans un délai acceptable.

La démonstration trouvée par l'ordinateur était à la portée des mathématiciens qui, comme le grand logicien polonais Alfred Tarski, s'étaient intéressés à la conjecture de Robbins. L'ordinateur n'a pas produit une démonstration trop longue pour être comprise par un humain, il a trouvé une démonstration difficile et contournée, mais d'une longueur raisonnable et accessible à l'entendement humain. Cette démonstration est d'ailleurs plus courte que d'autres que les mathématiciens écrivent et échangent ordinairement, et qui atteignent parfois plusieurs centaines de pages.

L'ordinateur n'est donc ici qu'un intermédiaire que l'on peut oublier quand le travail est terminé. Notons aussi que, depuis le calcul d'octobre, des preuves un peu plus courtes ont été trouvées, ainsi que des démonstrations automatiques que l'équation de S. Winker suffit pour rendre «de Boole» une algèbre «de Robbins». Par prudence, toutes ces preuves ont été vérifiées par des programmes indépendants. Les données des démonstrations et des vérifications sont accessibles à tous par le réseau *Internet* ; c'est d'ailleurs ce qui m'a évité d'avoir à saisir à la main la démonstration indiquée à la figure 4 : il m'a suffi de la télécharger, supprimant ainsi les risques d'erreur de saisie.

Tout cela permet donc d'affirmer sans l'ombre d'un doute : les algèbres de Robbins sont des algèbres de Boole.

RÉSULTATS SIMPLES, PREUVES COMPLEXES

En examinant la preuve (même sans en regarder le détail), on est étonné de ce qu'un résultat si simple nécessite un si long détour. Naïvement, on est tenté de croire que tout ce qui est simple doit se démontrer simplement. L'exemple de la conjecture de Robbins, comme bien d'autres en mathématiques, illustre que ce n'est pas vrai. Les recherches systématiques de l'ordinateur établissent indirectement que le résultat ne possède aucune démonstration élémentaire, sinon elle aurait été trouvée. Le résultat est simple, mais le chemin invraisemblablement détourné qui y mène ne possède pas de raccourcis importants (ceux trouvés depuis octobre n'améliorent que marginalement la première preuve).

Des résultats de logique établissent d'ailleurs que, dans toute théorie intéressante, le rapport entre la longueur de la plus courte preuve d'un théorème et celle de son énoncé n'est pas limité

5. DÉMARCHE À SUIVRE POUR LES DÉMONSTRATIONS AUTOMATIQUES

PRÉPARATION

- Ramener le problème à un problème parfaitement clair et formalisé
- Rechercher un démonstrateur adapté, en modifier un ou en concevoir un nouveau (et en faire l'étude mathématique)
- Tester le démonstrateur et en ajuster les paramètres

DÉMONSTRATION

- Lancer le démonstrateur et attendre qu'il trouve une preuve (ou l'arrêter s'il ne trouve rien) ;
- Sauvegarder les traces informatiques de la démonstration (on ne saurait se contenter d'un «oui c'est vrai je l'ai prouvé» donné par l'ordinateur!)

VÉRIFICATION

- Faire redécouvrir la preuve par un second démonstrateur en le guidant éventuellement pour lui rendre la tâche plus facile, ce qui permet – même si le second démonstrateur n'est pas assez puissant pour trouver la preuve seul – de contrôler que le premier n'a pas commis d'erreurs ;
- ou bien produire une version courte et en notation simple de la preuve et la confier à un vérificateur de preuves sachant traiter ce type de formalismes ;
- ou bien refaire le calcul à la main à partir des traces informatiques laissées par le démonstrateur ;
- Mettre le tout sur *Internet* à la disposition des autres chercheurs, qui pourront contrôler à nouveau ;
- Publier la preuve dans les revues spécialisées, ce qui permettra d'autres contrôles.

en taille : pour certains résultats, ce rapport sera de 10, pour d'autres de 1000, pour d'autres de 1000^{1000} , etc. Ces propriétés sont liées à l'indécidabilité logique qui est bien présente dans le problème de Robbins, comme l'a démontré le mathématicien G. McNulty en 1976 : aucune méthode algorithmique générale ne permet de décider à coup sûr, pour tout système d'axiomes, s'il est équivalent ou non aux axiomes d'une algèbre de Boole.

CE N'EST ENCORE QUE DU CALCUL

Reconnaissons-le, à voir la preuve donnée le programme de W. McCune, on remarque que, encore une fois, le démonstrateur automatique a essentiellement fait un calcul : il a accumulé des équations, certaines très compliquées, il a tenté de les combiner de nombreuses façons, et a fini par aboutir, sans qu'aucune compréhension nouvelle ne se déduise de sa démonstration. Nous sommes donc encore déçus. De surcroît, peu de domaines se prêtent aujourd'hui aussi bien que les algèbres de Boole à l'utilisation de démonstrateurs de théorèmes.

Les résultats nouveaux découverts par des programmes de démonstration automatique sont semblables à la

conjecture de Robbins : ils font intervenir des formules compliquées que le programme manipule avec obstination, lui permettant d'aboutir là où aucun humain n'a le courage et la patience de s'aventurer.

Malheureusement (ou heureusement?), les méthodes de démonstration automatique sont ainsi peu susceptibles de fournir des résultats nouveaux intéressant les mathématiciens ; ce n'est que sur des questions très particulières que, pour l'instant, ils ont fait avancer la recherche.

LES DÉMONSTRATEURS DE THÉORÈMES

Pour justifier les travaux en démonstration automatique, nous allons passer d'autres arguments en revue. Ils établissent sans discussion possible l'intérêt de ce qui a été fait dans cette discipline informatique depuis 1954, date à laquelle le premier démonstrateur de théorèmes a été écrit par Martin Davis pour établir des résultats élémentaires d'arithmétique.

D'abord, les démonstrateurs de théorèmes et les méthodes qu'on en tire sont utiles en dehors des mathématiques. Prouver la correction d'un circuit, d'un programme ou d'un protocole d'échange de données n'intéresse pas les mathématiciens, mais est capi-

tal pour les sociétés qui produisent des microprocesseurs ou dans le domaine de la fiabilité des programmes. Aujourd'hui, les industriels utilisent des logiciens dérivés de ces programmes de démonstration automatique. Le Centre national d'études spatiales, le Centre national d'études des télécommunications et l'Aérospatiale, en relation avec des universitaires et des chercheurs du CERT-ONERA de Toulouse, par exemple, mettent en œuvre de tels développements.

En intelligence artificielle, les logiciens systèmes experts qui tentent d'égaliser les experts de domaines particuliers (médecine, détection et identification de pannes, conseil en placements boursiers, etc.) comportent des mécanismes de raisonnement appelés *moteurs d'inférences* qui proviennent en grande partie des recherches en démonstration automatique. De même, les langages d'interrogation des *bases de données* exploitent, pour répondre aux requêtes des utilisateurs, des mécanismes de calcul issus de la démonstration automatique.

Le langage de programmation *Prolog*, inventé à Marseille par Alain Colmerauer et Philippe Roussel en 1972, est très prisé pour le développement d'applications en intelligence artificielle. Il est conçu autour d'un mécanisme tiré d'un démonstrateur de théorèmes. La puissance de ce langage provient de cette origine et en explique la caractéristique principale : écrire un programme *Prolog* ne consiste pas à donner des ordres à l'ordinateur (principe de la programmation classique, dite *impérative*), mais à décrire un problème que le langage se chargera de résoudre (principe de la programmation *déclarative*).

De plus, en mathématiques, les démonstrateurs de théorèmes sont souvent les composants de base des vérificateurs de preuves.

VÉRIFIER SANS RÉPIT

Les erreurs en mathématiques sont fréquentes et, parfois, passent inaperçues de longues années. Dans le cas du théorème des quatre couleurs, par exemple, des mathématiciens ont publié à deux reprises des preuves incorrectes, et deux fois l'erreur est restée ignorée durant plusieurs années. Les exemples de démonstrations fausses sont innombrables en mathématiques. On peut penser que les parties centrales des mathématiques sont exemptes de telles erreurs, car leurs résultats ont été contrôlés par de nombreux mathématiciens, et ont été recoupés. En revanche, dans les domaines avan-

cés de recherche où peu de mathématiciens travaillent, une erreur peut rester ignorée longtemps, voire indéfiniment si le résultat faux, trop en marge, n'est jamais utilisé. Les démonstrations fausses ou incomplètes de résultats justes, quoique moins ennuyeuses, sont aussi légion, et leur détection est encore plus délicate.

Il est naturel de vouloir être certain que les preuves mathématiques proposées par les mathématiciens sont bonnes. Cela pourrait être accompli en les soumettant au verdict impartial des machines. Cette idée a déjà conduit à la vérification de théorèmes mathématiques classiques.

Le projet AUTOMATH de N.G. de Bruijn, dès les années 1970, validait une partie du célèbre livre sur les fondements de l'analyse du mathématicien allemand Edmund Landau.

D'autres projets ont permis la vérification de preuves non triviales dont voici une liste très partielle :

- le théorème de Cantor sur le cardinal de l'ensemble des parties d'un ensemble (et qui indique qu'il y a plusieurs types d'infinis) ;
- le théorème des valeurs intermédiaires en analyse (qui dit qu'une fonction d'une variable continue qui prend les valeurs a et b prend aussi toutes les valeurs entre a et b) ;
- le théorème du point fixe de Banach (qui indique que, lorsque vous froissez une feuille de papier et la remettez dans le périmètre qu'elle occupait avant, alors, au moins un point est exactement au dessus de l'endroit qu'il occupait auparavant) ;
- le théorème de Wilson en arithmétique (qui stipule que p est un nombre premier, si et seulement si $(p-1)!+1$ est divisible par p) ;
- l'indécidabilité de l'arrêt d'un programme, démontrée en 1936 par Turing.

En France, le projet COQ propose ainsi un système puissant pour la formalisation et la vérification de démonstrations dans des domaines variés des mathématiques. Malgré ces succès obtenus et les incontestables progrès de ces dernières années, reconnaissons que peu de mathématiciens utilisent les programmes mis à leur disposition pour vérifier leurs démonstrations.

C'est que ceux-ci ne sont pas assez commodes, pour ce que le mathématicien ordinaire produit quotidiennement : les étapes de codage et de formalisation pour obtenir la vérification par l'ordinateur des résultats mathématiques nécessitent une patience et une familiarité qui, finalement, les réservent à quelques spécialistes. L'utilité

de ces systèmes d'aide aux mathématiciens ne fait pourtant aucun doute, car ils permettraient de valider plus rapidement les preuves produites, d'en dériver des versions testables sur d'autres systèmes (vérifier une preuve plusieurs fois ne consisterait plus à la lire, mais à la faire passer dans des vérificateurs automatiques différents), et libérerait le mathématicien de certaines parties pénibles de son travail.

C'est pourquoi un projet international appelé QED (des trois mots *quod erat demonstrandum*, qui signifient «ce qu'il fallait démontrer») tente de se mettre en place depuis 1994. Son but est très ambitieux, et ses promoteurs le comparent à la rédaction du traité général de mathématiques de Bourbaki entreprise il y a quarante ans en France. Il s'agit de faire coopérer tous les spécialistes du domaine, de façon à accélérer les progrès.

La capacité à découvrir de nouvelles démonstrations non triviales et à vérifier celles produites par les mathématiciens a atteint un niveau intéressant ; d'après les tenants du projet, tous les mathématiciens l'utiliseront d'ici peu.

Avec le succès récent de *Deeper Blue* sur Gary Kasparov, le 11 mai 1997, dans une compétition en bonne et due forme, et malgré les commentaires sceptiques qu'il a provoqués (voir *Pour la Science*, juin 1997), nous assistons à la naissance d'une intelligence artificielle aux capacités spécialisées, mais dont on ne peut plus dire qu'elle échoue dans tout ce qu'elle entreprend.

Jean-Paul DELAHAYE est directeur adjoint du Laboratoire d'informatique fondamentale de Lille du CNRS.
e-mail : delahaye@lifl.fr

W. McCUNE et R. PADMANABHAN, *Automated Deduction in Equational Logic and Cubic Curves*, Lecture Notes in Computer Science n° 1095, Springer-Verlag, ISBN 3-540-61398-6, 1996.

C.W.H. LAM, *How Reliable Is a Computer-Based Proof? in The Mathematical Intelligencer*, vol. n° 12, pp.8-12, 1990.

M. RUSINOWITZ, *Démonstration automatique. Techniques de Réécriture*, InterEditions, Paris, 1989

T. L. SAATY et P. C. KAITEN, *The Four-Color Problem, Assaults and Conquests*, Dover Publications, Inc. New York, 1977, 1986.

L. WOS, *The Automation of Reasoning : An Experimenter's Notebook with Otter Tutorial*, Academic Press, New York, 1996.
