Les ordinateurs mathématiciens !

Jean-Paul DELAHAYE

(avec l'aimable autorisation de la revue "Pour la Science"; cet article est paru dans le n° 239 de septembre 1997)

Après le jeu d'échecs, l'intelligence artificielle triomphera-t-elle des mathématiques ?

II y a 40 ans que les calculatrices ont supplanté l'homme pour les calculs numériques. En revanche, c'est seulement depuis une dizaine d'années que le calcul symbolique par ordinateur (dérivation des fonctions, recherche de primitives, manipulation de polynômes, développements limités, etc.) a atteint une maturité qui en fait un allié de première importance dans certains domaines mathématiques et une aide appréciée des ingénieurs. La nouvelle formule de calcul de n, que nous avons traitée en janvier 1997, était un exemple frappant de cette avancée et des résultats produits par le calcul symbolique en mathématiques.

Un domaine mathématique subsiste où les progrès de la science informatique sont peu spectaculaires : la recherche de démonstrations par ordinateur.

Distinguons deux types de démonstrations mathématiques : celles qui ne sont que le développement de longs calculs (numériques ou symboliques) et celles qu'on obtient avec des méthodes où les raisonnements prédominent.

Pour le second type de démonstration, l'apport de l'informatique n'est pas convaincant : rares sont les énigmes anciennes intéressant vraiment les mathématiciens dont la résolution a été obtenue par une démonstration automatique d'ordinateur.

Le théorème des quatre couleurs

On cite fréquemment le théorème des quatre couleurs : avec quatre couleurs, on peut toujours colorier les pays d'une carte géographique sans que deux voisins portent des couleurs identiques. Ce résultat, conjecturé en 1852 par Francis Guthrie, ne fut démontré qu'en juin 1976 par Kenneth Appel et Wolfgang Haken à l'aide d'un calcul par ordinateur. Toutefois, les idées et la structure de la démonstration étaient dues aux mathématiciens qui, pendant plus d'un siècle, avaient travaillé sur le problème. L'ordinateur, en 1976, s'est chargé d'un travail de recherche et de vérification trop long pour être mené à la main par un humain, mais qu'on peut difficilement considérer comme un raisonnement et encore moins comme un raisonnement créatif. L'ordinateur, dans la démonstration du théorème des quatre couleurs, quoique irremplaçable, n'a été qu'un assistant fiable et obstiné.

La preuve de K.Appel et W.Haken n'a jamais été vérifiée à la main. En fait, elle n'avait pas été vérifiée du tout, car, pour le faire par machine, il fallait saisir la description de 1 476 graphes assez compliqués, ce qui, à cause du risque considérable d'erreurs et de la stupidité de cette étape de travail, n'a pas été tenté.

Neil Robertson, Daniel Sanders, Paul Seymour et Robin Thomas entreprirent, il y a quelques mois, de trouver une nouvelle preuve du théorème des quatre couleurs, afin de tranquilliser les mathématiciens qui niaient l'authenticité de la démonstration : en mathématiques, un résultat démontré une seule fois - par un humain ou par un ordinateur - reste incertain.

Après quelques mois de travail, l'équipe réunie autour de W. Robertson et leur ordinateur découvrirent une nouvelle preuve, plus courte que la précédente. Elle utilise un ensemble de 633 graphes, dont l'un au moins serait présent dans un contre-exemple au théorème des quatre couleurs, s'il en existait.

La méthode est en gros la même que celle de 1976, et la nouvelle preuve est encore impossible à vérifier à la main. Toutefois, avoir trouvé deux preuves différentes du théorème devrait rassurer les mathématiciens soupçonneux. De plus, puisque la nouvelle preuve est plus courte, elle sera aussi plus facile à vérifier; d'autant qu'un effort a été fait pour rendre disponible les données nécessaires à la vérification sous une forme commode à manipuler et accessible à tous par Internet

La communication généralisée des ordinateurs facilite l'échange des données et, donc, la vérification de ce type de résultats par d'autres équipes, tâche autrefois presque impossible.

LE PLAN PROJECTIF D'ORDRE 10

Un second exemple de résultat souvent mentionné comme une réussite des ordinateurs en mathématiques est la preuve qu'il n'existe pas de plan projectif fini d'ordre 10 (une explication de ce qu'est un plan projectif et un exemple sont indiqués sur la figure 2). Cette preuve a été obtenue à l'aide d'un programme, en 1988, par C.W.Lam, L.H. Thiel et S.Swiercz.

Le problème est intéressant, car il est lié à une conjecture formulée en 1782 par Léonard Euler concernant les carrés gréco-latins, qui, une fois qu'elle eut été montrée fausse (en 1959), amena à penser qu'il pouvait y avoir de tels plans projectifs d'ordre 10. L'échec des tentatives pour en construire ou pour démontrer qu'ils sont impossibles faisait de la question une énigme agaçante que l'informatique a résolue. II s'agit donc, là encore, d'un résultat sur lequel les mathématiciens s'interrogeaient (une thèse a même été soutenue sur le sujet à l'Université de Berkeley en 1974) et qui a été trouvé par ordinateur avant qu'aucun humain ne l'élucide.

L'ordinateur fut irremplaçable, mais son rôle fut d'explorer systématiquement et mécaniquement toutes les tentatives imaginables de construction d'un plan projectif d'ordre 10 et de constater qu'aucune n'aboutit. Le travail fait par l'ordinateur se réduisait à un calcul.

Comme pour le théorème des quatre couleurs, une telle preuve est, par sa longueur, impossible à vérifier à la main.

II y a quelques mois, William McCune, du Département de mathématiques et d'informatique du Laboratoire Argonne, aux Etats-unis, a démontré la conjecture de Robbins, sur laquelle des recherches vaines étaient menées depuis 1933.

L'ordinateur cette fois-ci, a-t-il fait un peu plus que dans les deux cas précédents? Pour que chacun puisse 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 XIXe 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*. II 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. II 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'équation 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.

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 en taille: pour certains résultats, ce rapport sera de 10, pour d'autres de 1 000, pour d'autres de $1\ 000^{1\ 000}$, 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 capital pour les sociétés qui produisent des microprocesseurs ou dans le domaine de la fiabilité des programmes. Aujourd'hui, les industriels utilisent des logiciels 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 oeuvre de tels développements.

En intelligence artificielle, les logiciels systèmes experts qui tentent d'égaler 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 avancés 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.

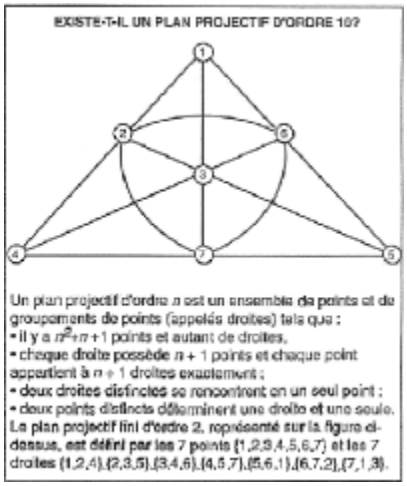
Avec le succès récent de Deeper-Blue sur Gary Kasparov, le 11 mai 1997, dans une compétition en bonne et due forme, et malgré les commentaires sceptiques qu'il a provoqués (voir *Pour la Science*, juin 1997), nous assistons à la naissance d'une intelligence artificielle aux capacités spécialisées, mais 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 ls a Computer-Based Proof ? in The Mathematical Intelligencer, vol. n° 12, pp.8-12, 1990.
- M. RUSINOWITC, 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.

Figure 2 :



Les mathématiciens ne pouvaient déterminer s"il existait un plan projectif d'ordre 10. Cette inexistence a été démontrée en 1988 sur un ordinateur programmé par C. Lam, L. Thiel et S. Swiercz.

Figure 3 :

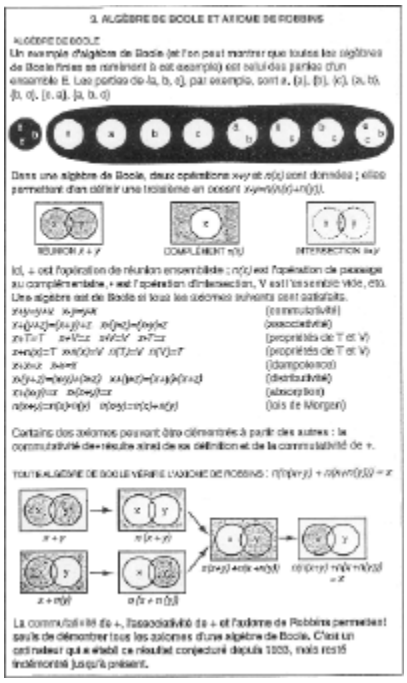


Figure 4

4. LA DÉMONSTRATION DU 2 OCTOBRE 1996

Voici la preuve de la conjecture de Robbins trouvées 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 existé *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] indiqué 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(n(x)+y+n(x+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(x)+y)+x+y+y)+n(n(x)+y)) = y

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

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

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

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

Figure 5

5. DEMARCHE A SUIVRE POUR LES DEMONSTRATIONS AUTOMATIQUES.

PREPARATION

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

DEMONSTRATION

- 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 " ou c'est vrai je l'ai trouvé " donné par l'ordinateur !)

VERIFICATION

- Faire redécouvrir !a 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 â un vérificateur de preuves sachant traiter ce type de formalismes ;
- ou bien refaire !e 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 !a preuve dans les revues spécialisées, ce qui permettra d'autres contrôles.