par L.T.D. Nguyen

La démonstration automatique : un enjeu crucial

S'assurer automatiquement qu'une proposition mathématique est vraie est, en théorie, impossible en général. Pourtant, dans certains cas, ça marche très bien en pratique !

Woodrow Wilson Bledsoe (1921–1995).

Qu'est-ce qu'une « preuve par ordinateur » ? On peut s'appuyer sur la machine pour faire une partie du travail d'une démonstration rédigée par un humain, notamment des calculs mécaniques ; vérifier que des fautes de raisonnement ne se sont pas glissées dans une preuve formelle fournie à l'ordinateur par l'humain ; parvenir sans assistance à déterminer qu'une conclusion est vraie. Le premier cas a fait couler beaucoup d'encre depuis quarante ans (voir *les Mathématiques de l'impossible*, Bibliothèque Tangente 49, 2013). Le deuxième est pris en charge par des logiciels appelés *assistants de preuve* (voir les articles suivants dans ce dossier).

Hilbert et le problème de la décision

Le but d'un prouveur automatique de théorèmes est, à partir d'un énoncé mathématique, d'établir s'il est démontrable ou non à partir de certains axiomes. Ce problème a été posé en 1928 par David Hilbert, qui lui a donné le nom de *problème de la décision (Entscheidungsproblem)*, pour les énoncés de la logique du premier ordre, qui englobe les mathématiques courantes. Alonzo Church, et indépendamment son élève Alan Turing, ont prouvé en 1936 qu'il était impossible à résoudre par un algorithme. C'est l'acte fondateur de l'informatique théorique. Cependant, restreindre le problème permet de s'en sortir.

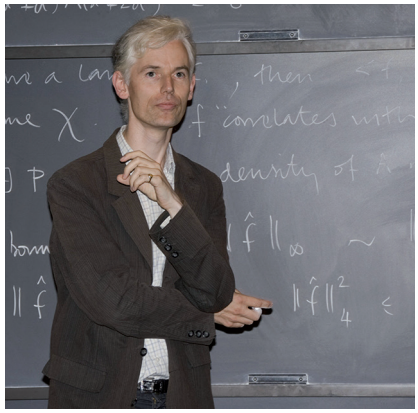
En logique propositionnelle, il s'agit de déterminer si une formule donnée est toujours vraie ; on dira alors qu'elle est *valide*. C'est le cas par exemple de « non (A et (non A)) ». Cela peut être résolu par un algorithme : on écrit la table de vérité de la formule et on regarde si toutes les lignes sont à « vrai ». Ce faisant, on se heurte rapidement au problème suivant : pour s'assurer que « non (A et (non A)) » est valide, il suffit de vérifier pour A vraie et pour A fausse. Pour $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$, il faut déjà regarder quatre combinaisons. Pour une formule faisant intervenir cent variables, le nombre de lignes de la table de vérité qu'il faudrait traiter est de 2^{100}, soit environ 10^{30}. C'est de loin hors de portée des ordinateurs dont on dispose ! Notre algorithme

n'est pas capable de répondre en un temps de calcul raisonnable.
Une meilleure technique est de rechercher un contre-exemple, c'est-à-dire un choix des valeurs de vérité des variables rendant fausse la formule. Les algorithmes qui procèdent de cette façon vont essayer de « deviner » les valeurs prises par une partie des variables dans un contre-exemple. En effet, souvent, lorsque certaines des variables sont fixées, la formule est vraie ou fausse indépendamment des valeurs prises par les autres. Dans le premier cas, on sait que ce n'est pas une bonne piste pour trouver un contre-exemple ; dans le second cas, on a trouvé notre contre-exemple. Lorsque l'on a épuisé tous les contre-exemples possibles, la formule est valide.

Un enjeu crucial

En pratique, ces méthodes (et bien d'autres) sont mises en œuvre dans des logiciels nommés *solveurs* SAT. Ceux-ci sont entièrement dédiés à la résolution du problème SAT, qui consiste à trouver un choix de valeurs de variables qui satisfassent une formule (autrement dit : qui la rendent vraie), ou d'indiquer si c'est impossible. Les solveurs les plus performants arrivent à résoudre en moins d'une seconde de nombreux cas comportant plusieurs centaines de variables ! Mais pas tous : SAT fait partie de la famille des problèmes NP-complets (voir l'article qui suit). Ce cadre très restreint pour la démonstration automatique conduit déjà à des applications en combinatoire ou en certification de circuits électroniques.

Une machine peut-elle raisonner comme un mathématicien humain ? Les ordinateurs peuvent prouver des théorèmes, mais en suivant des procédures de recherche qui n'ont rien à voir avec la façon dont un humain s'y prendrait pour attaquer le problème. Surtout, elles ne produisent pas de preuve lisible, qu'un humain pourrait facilement comprendre par lui-même. Les méthodes « humaines » pour la preuve automatique n'ont pourtant pas été complètement délaissées. Woody Bledsoe, pionnier de l'intelligence artificielle, s'y est intéressé dès les années 1970. Il a tenté d'incorporer le raisonnement par analogie en démonstration automatique. Plus récemment, l'idée de s'inspirer du raisonnement tel que pratiqué par les mathématiciens est également à la base de recherches de Timothy Gowers et Mohan Ganesalingam. Leur objectif : reproduire algorithmiquement les preuves qu'un humain aurait naturellement trouvées pour des problèmes simples, en partant d'une analyse linguistique du langage des mathématiciens.



© Charles Jeffery Mozzochi

Sir William Timothy Gowers.

Dans un monde où l'informatique prend de plus en plus d'importance, la certification des logiciels est devenue un enjeu crucial, et les problèmes algorithmiques liés à la démonstration sont au cœur des recherches sur le sujet. Il semblerait donc que la preuve automatique ait de beaux jours devant elle, malgré l'indécidabilité et la NP-complétude !

L.T.D. N.