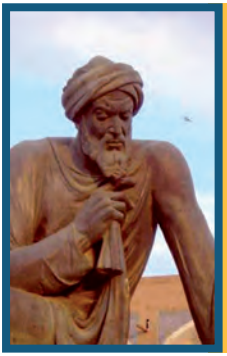
Informatique et mathématiques, un partenariat gagnant

Gérard Berry
Professeur au Collège de France,
membre de l'Académie des sciences
et de l'Académie des technologies,
médaille d'or 2014 du CNRS

Les mathématiques et l'informatique partagent deux ancêtres communs : le nombre et le calcul, vite devenus indispensables pour le comptage des troupeaux, la mesure du temps, celle des distances et des surfaces, l'architecture, l'observation des astres, le commerce, l'impôt… Les procédés de calcul sur les nombres datent de plusieurs milliers d'années pour la multiplication et la solution des équations simples. Devenus petit à petit systématiques et se faisant aider par des outils variés (abaques, bouliers…), ils permettaient de ne plus faire appel à l'intelligence pour réaliser les opérations : c'étaient déjà des *algorithmes* avant la lettre.



Le mot « algorithme » désigne un procédé de calcul systématique. Il vient du nom du savant persan al-Khuwarizmi (vers 783, 850), qui a importé les connaissances mathématiques des Indiens (dont les chiffres sont du coup devenus nos chiffres arabes).

Il a écrit le traité *Al Jabr* sur la solution des équations ; de là vient le mot « *algèbre* ».

Photo : Maurice Nivat

De la démonstration à l'ordinateur : une évolution « logique »

Les mathématiques telles que nous les connaissons et les pratiquons aujourd'hui sont arrivées plus tard. On les retrouve déjà chez les pythagoriciens pour justifier les algorithmes empiriques, par exemple pour montrer que $a^2 + b^2 = c^2$ est vrai quel que soit le triangle rectangle de

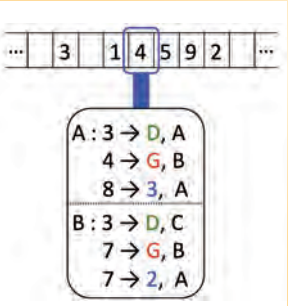
côtés quelconques *a*, *b* et *c*, et pas seulement pour les nombreux exemples connus. Il a fallu introduire la nouvelle notion de démonstration, clef de la pensée mathématique. Cette notion a ensuite été elle-même mathématisée par les logiciens, qui l'ont transformée en calcul : si inventer une démonstration est intellectuellement difficile, la vérifier doit être une opération algorithmique conceptuellement réalisables par une machine. Cette approche calculatoire de la logique a culminé dans le fameux théorème de Gödel, qui montre les limites du procédé en exprimant que toutes les vérités mathématiques ne sont pas démontrables par calcul logique.

Alan Mathison Turing (1912–1954) a grandement participé à la victoire des Alliés en cassant le code secret allemand Enigma pendant la Seconde Guerre mondiale.

L'informatique n'est vraiment née qu'en 1936, quand Alan Turing a défini sa fameuse machine (voir ci-dessous). Appliquant les idées de Gödel à cette construction ultra-minimale, Turing a démontré des résultats extraordinaires : d'abord, sa machine permet de faire tous les calculs dont on a besoin en pratique ; ensuite, on peut construire une machine universelle programmable, où un programme enregistré à côté des données définit les calculs à faire, la machine restant constante ; enfin, la machine universelle n'est pas omnipotente, car elle ne peut pas évaluer par le calcul si un programme précis va s'arrêter sur une donnée précise.

La machine de Turing

La machine conceptuelle mise au point par Turing comporte une tête de lecture, qui permet de lire et d'écrire des lettres appartenant à un alphabet fini sur une bande infinie découpée en cases, et de décaler la bande d'une case à gauche ou à droite. Elle détermine ses actions seulement à partir de la lettre lue sous la tête et d'un état fini géré par un contrôleur qui dit simplement quelle action faire :
« écrire une autre lettre »
ou
« bouger la bande d'une case ».



Paradoxalement, la machine de Turing possède les mêmes capacités (mais bien sûr pas la même efficacité) que tous les ordinateurs connus, et, selon la thèse de Church, que tous les procédés de calcul à venir.

L'ENIAC
a été le premier ordinateur.
Il pesait plusieurs tonnes
et avait un temps de fonctionnement
sûr de seulement quelques minutes.
Son programme était câblé,
pas encore enregistré en mémoire.
© US Army
(Encadré) Le microprocesseur
Haswell d'Intel comporte plus
d'un milliard de transistors.
Crédit : AnandTech

Mais il faudra attendre la fin des années 1940 pour que l'informatique passe à l'acte avec les premiers ordinateurs électroniques construits par des ingénieurs et des mathématiciens (dont John von Neumann), puis les inventions des transistors et des circuits intégrés modernes pour passer à l'échelle actuelle : si le calculateur ENIAC avait seize mille tubes à vide, un processeur moderne possède plusieurs milliards de transistors.

Le développement d'une pensée algorithmique propre

Les ordinateurs ont d'abord servi d'outils de calcul scientifique pour mathématiciens. Mais ils ont vite pénétré d'autres secteurs, remplaçant les machines mécanographiques dans la gestion des entreprises. Dès la fin des années 1960, avec le progrès exponentiel des ordinateurs, l'explosion des domaines d'application et l'accroissement constant de sa communauté, l'informatique a commencé à se détacher des mathématiques classiques pour développer sa propre *pensée algorithmique* : invention des structures de données et développement des algorithmes associés, conception de langages permettant une programmation plus riche et non limitée aux nombres, *etc*. Cette évolution a demandé l'élaboration de nouvelles approches mathématiques, comme en cryptographie [voir notre article en page 21], pour résoudre des questions… typiquement informatiques.

La *complexité des algorithmes* cherche à estimer le coût en temps, mémoire et énergie des calculs pour un problème donné. Elle a conduit à une classification fine de la difficulté intrinsèque des problèmes algorithmiques, et aussi à la fameuse conjecture « la classe P est-elle égale à la classe NP ? » (énoncée par Stephen Cook en 1972), qui fait partie de la liste des problèmes de maths mis à prix à un million de dollars.

P *versus* NP et SAT

La classe P est celle des problèmes solubles en temps polynomial, donc « rapidement » (en pratique). La classe NP est celle des problèmes pour lesquels il est « facile » de vérifier une solution (proposée par exemple par un oracle), mais pas de proposer la bonne solution. Il existe des centaines de problèmes pratiques dans ce cas !

Par exemple, le problème de la *k*-coloration de graphe cherche s'il est possible de colorer un graphe non orienté avec *k* couleurs de manière à ce que deux sommets reliés par un arc ne soient pas de la même couleur. Il est très facile de vérifier si une coloration a bien la propriété cherchée, mais très difficile d'en trouver une bonne.

Le problème NP le plus typique est SAT, le vieux problème de la satisfaction d'une formule booléenne (sur vrai / faux, ou 0 / 1) écrite avec des variables et les opérateurs et, ou, non. Une telle combinaison de variables booléennes étant donnée, savoir s'il existe des valeurs (0 ou 1) de ces variables qui rendent l'expression globale égale à 1 est un problème NP-complet, ce qui veut dire que tout autre problème NP peut être réduit à lui. La conjecture est que SAT ne peut pas être résolue en temps polynomial dans le pire cas. Mais cela ne veut pas dire qu'il ne peut pas être résolu dans les cas pratiques ! Avant 2000, on ne pensait pas pouvoir résoudre SAT au-delà de quelques centaines de variables. Avec de nouveaux algorithmes étonnants, on résout maintenant des problèmes à deux millions de variables, y compris en cadre industriel !

La *théorie de l'information* cherche quant à elle à comprendre comment transmettre efficacement des messages sur des lignes bruitées. Créée par Claude Shannon en 1947, elle a permis des gains de facteur 10 000 entre la transmission par modems des années 1980 et l'ADSL actuel sur les mêmes fils téléphoniques, et des gains tout aussi considérables pour la 5G en téléphonie sans fil. Information et probabilités sont liées. La transmission en télécoms, le routage des paquets sur Internet et la gestion des réseaux pair-à-pair font appel à des algorithmes probabilistes sophistiqués. En fait, le hasard et les probabilités jouent un rôle de plus en plus important en algorithmique.

Géométrie, théorie des nombres, analyse : toutes les maths mobilisées !

La *géométrie algorithmique* cherche à développer des algorithmes très efficaces pour les grands calculs géométriques nécessaires pour la CAO (conception assistée par ordinateur) des avions, voitures, ou tout autre objet, pour la synthèse d'images, pour la modélisation des molécules et de leurs interactions…

Le *traitement d'images et de sons* a pris une importance considérable, avec des résultats insoupçonnables auparavant. Par exemple, au lieu de regarder séparément des images par radio, scanner et IRM, le médecin va regar-

der désormais des *fusions algorithmiques* de ces images sous formes 3D ou 4D (films), qui lui fourniront des possibilités de diagnostic bien plus fines.

La *modélisation* et la *simulation numériques* de phénomènes complexes se développent dans de nombreuses branches : médecine, avionique, motorisation, astrophysique, agronomie, biologie. On commence même à faire réaliser des calculs destinés à agir sur des réseaux métaboliques par une *machine biochimique intra-cellulaire*, qui remplace les transistors par des brins d'ADN…

Le vieux rêve de Turing commence à se réaliser

N'oublions pas le rôle majeur de la logique mathématique dans l'informatique moderne ! Les premiers constructeurs d'ordinateurs ont constaté, à leur grande surprise, qu'il était très difficile d'écrire des programmes justes. Ils ont dû inventer le *débogage,* activité apparemment fort éloignée des mathématiques. Ici encore, Turing a été le grand précurseur, dans une note de 1949 exprimant que la meilleure façon de savoir si un programme fait ce que l'on veut serait de le vérifier mathématiquement. Il a proposé des solutions à base d'*assertions logiques* placées sur les lignes de programmes et d'*invariants mathématiques* permettant de prouver ces assertions. Après une lente évolution (qui a cependant donné lieu au prix Turing du Français Joseph Sifakis en 2006), la vérification de programmes est devenue mature. Elle se décompose en *vérification automatique* pour des sous-domaines dédiés (circuits électroniques…) et *vérification par assistants de preuves mathématiques.* Les deux ont fait des progrès extraordinaires ces quinze dernières années, avec des succès pratiques comme la vérification systématique de la conception des circuits électroniques, celle du compilateur C CompCert de Xavier Leroy ou celles de protocoles de sécurité pour la carte bleue. Le vieux souhait de Turing se réalise !

Si les mathématiciens utilisent depuis longtemps les systèmes informatiques de calcul formel pour gérer leurs gros calculs, l'utilisation d'assistants informatiques de preuve pour vérifier les théorèmes est plus récente. Les plus puissants sont fondés sur le λ-calcul (« lambda-calcul »), introduit en 1936 par Alonzo Church en même temps que la machine de Turing, et devenu la base des langages de programmation et des logiques modernes. Le système français Coq, leader actuel du domaine, résulte de trente années de recherches sur le λ-calcul et la théorie des types par Jean-Yves Girard, Gérard Huet, Thierry Coquand et bien d'autres.

En utilisant Coq, Georges Gonthier et Benjamin Werner ont démontré en machine le fameux théorème des quatre couleurs (voir en encadré). Puis Gonthier et son équipe ont réalisé un exploit exceptionnel, vérita-

blement historique, en démontrant en Coq le célèbre théorème de Feit–Thompson (datant de 1963) sur la classification des groupes d'ordre impair. La preuve publiée de ce difficile théorème d'algèbre occupe pas moins de deux cent cinquante pages de maths lourdes, que l'on n'a jamais vraiment réussi à simplifier de manière significative. Enfin, en 2014, Thomas Hales a fourni la première démonstration de la fameuse conjecture de Kepler sur le rangement de sphères toutes identiques, qui dit simplement que les épiciers ont raison de mettre leurs oranges plan par plan sous forme d'hexagones. La preuve de Hales a demandé d'une part une utilisation massive du calcul formel sur les polynômes, et d'autre part des vérifications assistées de preuves en utilisant le système anglais HOL (Higher Order Logic).

Le théorème des quatre couleurs

Le *théorème des quatre couleurs* exprime qu'il suffit de quatre couleurs pour colorier toute carte de géographie de façon à ce que deux États avec une frontière commune n'aient pas la même couleur. La question de savoir si toute carte peut être ainsi coloriée a été posée en 1852. En 1976, Appel et Haken rédigent un article montrant qu'il suffisait de vérifier 1 478 configurations de cartes, ce qu'ils font sur ordinateur. Leur article n'est pas contesté, mais la vérification sur ordinateur n'est pas vue comme une « vraie » preuve. En 2006, à l'aide du système Coq, Gonthier et Werner prouvent réellement le théorème, simplifient la preuve, et montrent que, si les vérifications sur ordinateur de 1976 étaient bien correctes, le texte mathématique original avait des preuves pas toujours complètes et devait être largement repris !

À la suite de ces exploits informatico-mathématiques, le fameux Institute For Advanced Study de Princeton (États-Unis) a dédié une année spéciale à la mécanisation des preuves mathématiques en Coq et poursuit ce travail dirigé notamment par le médaillé Fields Vladimir Voïevodsky. Aux États-Unis toujours, la National Science Foundation a créé un grand projet, The Science Of Deep Specification, sur la vérification

G.B.

Pour en savoir (un peu) plus :

Pourquoi et comment le monde devient numérique. Gérard Berry, Fayard–Collège de France, 2008.

Vidéos et supports des cours de 2008 à 2016 au Collège de France : ***college-de-france.fr/site/gerard-berry***.

Les métamorphoses du calcul. Gilles Dowek, Le Pommier, 2007.

Une histoire illustrée de l'informatique. Pierre-Étienne Mounier-Kuhn et Emmanuel Lazard, EDP Sciences, 2016.