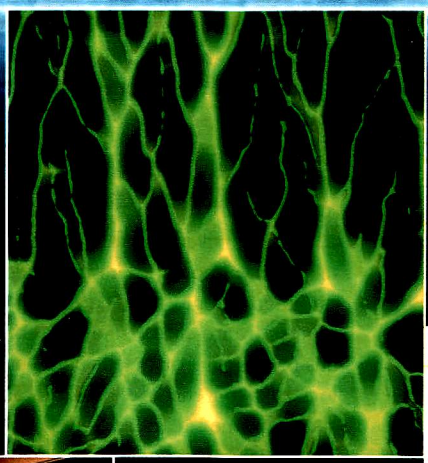
La vraie nature

YUTAKA EBEHARA, PHOTONICA / LARRY EDLING POUR SCIENCE & VIE - AFP / PASIEKA SPL COSMOS

PAR HERVÉ POIRIER

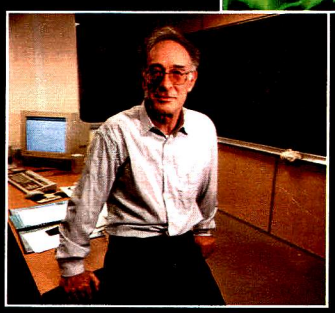
C'est peut-être l'une des plus importantes découvertes de tous les temps ! En exploitant jusqu'au bout une étrange correspondance entre les mathématiques et l'informatique, un réputé logicien français bouleverse radicalement notre vision du cerveau. Grâce à lui, nous savons peut-être enfin d'où vient l'intelligence ! D'ores et déjà, sa théorie éclaire d'un jour nouveau nombre de processus cognitifs. Et révolutionne totalement l'informatique.

de l'intelligence



Toute pensée est un calcul

Dans le cerveau se superposent des couches de langages, dont le premier est le lambda-calcul. Une théorie révolutionnaire. *p.40*



Au-delà des maths

Un travail colossal : la théorie de Jean-Louis Krivine marque l'aboutissement d'un siècle de recherches en logique. Les secrets d'un mathématicien de génie. *p.50*

L'informatique sauvée des bugs

C'est la solution pour débarrasser les logiciels de leurs bugs les plus retors : rendre les programmes aussi indubitables que les théorèmes mathématiques. Une correspondance que permet le lambda-calcul... *p.54*

Toute pensée est un calcul

En élaborant pour la première fois une architecture globale du cerveau, une théorie révolutionne ce que nous savons de l'intelligence, mais aussi l'informatique, la linguistique... et jusqu'aux rêves ! Et ce n'est qu'un début.

Il n'y a pas si longtemps, l'homme s'imaginait au centre de l'Univers. Vint Galilée, et tout fut à revoir... De tels moments sont rares dans l'histoire. Or, nous sommes peut-être en train de vivre l'une de ces révolutions. De celles qui commencent sans bruit pour n'avoir ensuite que plus de retentissements. De celles où c'est, pour une fois, notre regard qui est interrogé, plutôt que ce qu'il voit, ou croyait, voir jusque-là. Et tout se découvre sous un nouveau jour.

Des exemples ? Les mathématiciens se croient en quête de vérités idéales : ils ne font que plonger dans les méandres de leur propre cerveau. Les informaticiens s'imaginent les pionniers d'un nouveau monde : ils n'ont de cesse de reproduire un processus en œuvre depuis des millions d'années. Les peuples se targuent d'avoir forgé des langues originales : elles ont toutes une structure commune. Nous imaginons avoir des pensées : nous ne faisons que du calcul. L'intelligence est un mystère : oui, mais son siège ne se trouve pas au niveau des neurones. Incroyable ? Et pourtant...

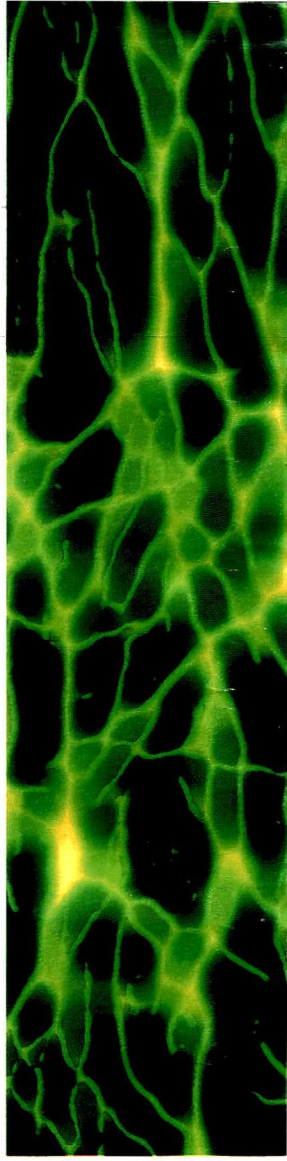
UNE VIEILLE INTUITION

- **IXe siècle** al-Khwarizmi publie les premiers algorithmes.
- **Fin du XVIIe siècle** Le philosophe Wilhelm Leibniz cherche «l'alphabet des pensées humaines».
- **1900** David Hilbert propose d'axiomatiser et de formaliser les mathématiques.
- **1901** Edmund Husserl veut fonder les vérités logiques sur des processus psychiques avec la phénoménologie.
- **1932** Alonzo Church invente le lambda-calcul.
- **1944** John von Neumann conçoit le premier ordinateur.
- **1949** Haskell Curry voit dans le lambda-calcul la structure logique sous-jacente aux langues naturelles.

LA CLÉ DU LAMBDA-CALCUL

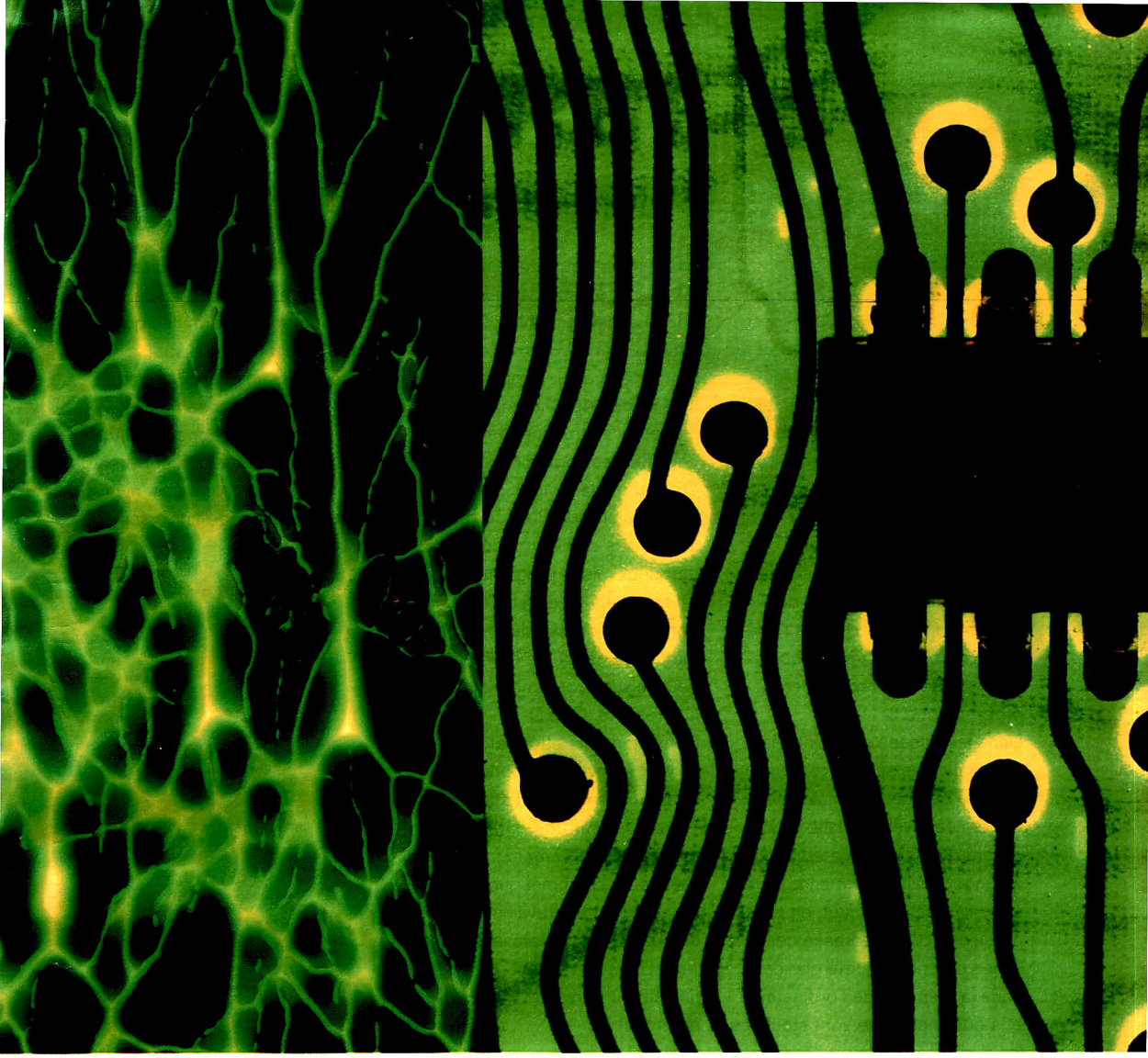
Pour en arriver là, il fallait une découverte. Elle a un nom : le "lambda-calcul", soit un langage logique mis au point dans les années trente et utilisé depuis cinquante ans par les informaticiens pour écrire leurs logiciels (voir encadré, p. 42). Il fallait aussi un homme : Jean-Louis Krivine, un mathématicien français de renommée mondiale. Et, surtout, une intuition : voir dans le lambda-calcul la structure logique qui régit l'intérieur de notre crâne, le langage universel émergeant du réseau de neurones qui grésillent dans notre cerveau, comme le comportement intelligent d'une fourmilière émerge des innombrables actes individuels et stupides des fourmis.

Dès lors, c'est toute l'architecture de notre cerveau qui prend une tournure inédite. Pour comprendre, il faut imaginer qu'au plus bas niveau s'enchevêtre l'inextricable flux de neurones, duquel émerge-



L'analogie parfaite

A gauche, les neurones du cerveau ; à droite, les filaments électriques d'un microprocesseur. Qu'il s'agisse du cerveau ou d'un ordinateur, l'architecture se fonde, au plus bas niveau, sur d'inextricables réseaux enchevêtrés. Le cerveau ressemble-t-il alors un ordinateur ? En réalité, ce serait le contraire. Et cela change tout...



A. PASIEKA/SPL/COSMOS

rait, à un niveau supérieur, un premier langage structuré : le lambda-calcul. Lequel serait lui-même représenté, à un niveau supérieur, par des couches de langages de plus en plus évolués, dont la plus haute correspond aux langues naturelles que nous parlons. Dans notre cerveau se superposeraient donc des couches de langages de plus en plus sophistiqués. Et l'intelligence ne résiderait pas au niveau de la grouillante activité neuronale, de même qu'il paraît vain de comprendre le fonctionnement d'un ordinateur en observant le mouvement des électrons qui s'enchevêtrent dans les circuits des puces électroniques. Au vrai, toutes nos pensées, conscientes et inconscientes, s'élaboreraient au niveau des couches de langages intermédiaires. Celles, précisément, au-dessus du lambda-calcul.

C'est un changement radical de perspective ! Car, en intercalant son point de vue entre la couche inférieure des neurones et les couches supérieures des langages évolués, la théorie de Jean-Louis Krivine va directement à l'essentiel. Située au point d'intersection du magma et du ciel, sur la terre ferme du lambda-calcul, elle permet, *in fine*, de dévoiler l'origine de toute activité cognitive, de saisir le secret même de l'intelligence. Derrière nos pensées mouline le lambda-calcul. D'ores et déjà, les conséquences sont... incalculables.

Ainsi, les mathématiciens sont persuadés, depuis trois mille ans qu'ils élaborent des

suite p. 43

Lambda-calcul
Les secrets d'un langage universel

Le lambda-calcul est un langage inventé en 1932 par le mathématicien américain Alonzo Church pour résoudre des questions de logique pure. Les mots de ce dialecte sont des formules de calcul. Trois opérations grammaticales sont autorisées : l'opération d'application, qui consiste à appliquer une formule f à une autre formule x, notée f(x) ; l'opération d'abstraction, qui consiste à baptiser une formule déjà construite, notée λxf(x) ; et l'opération de bêta-réduction, qui consiste à exécuter le calcul proprement dit.

Dans les années cinquante, les informaticiens se sont rendus compte que cette syntaxe minimaliste se révéle parfaite pour programmer un ordinateur. Lorsqu'un programme est enregistré sur une machine, il faut, en effet, préciser à quelle adresse ces lignes de code ont été gravées dans la mémoire de l'ordinateur, et à quelle adresse sont stockées les données que le programme doit analyser. C'est exactement ce que permet le lambda-calcul, si l'on remplace la notion de formule par celle de programme. Dès lors, pour un programme P, l'opération d'application P(x) précise l'adresse x où se trouvent les données ; l'opération d'abstraction λxP(x) définit l'adresse où est stocké le programme lui-même ; et la bêta-réduction fait tourner le programme.

Vu ainsi, il n'est pas étonnant que le lambda-calcul apparaisse comme un langage de programmation universel.

Il est plus surprenant, par contre, de constater qu'il permet aussi de définir les objets mathématiques. Ainsi, le nombre 2 est le lambda-terme λPλxP(P(x)). Une petite explication s'impose ici : selon l'idée de Church, l'entier 2 doit être l'instruction « répétez deux fois le programme qui suit ». Or, d'après la lambda-grammaire, l'expression λxP(P(x)) correspond à l'adresse de l'instruction répétant deux fois le programme P. En rajoutant λP devant, on tombe donc sur l'adresse du programme qui demande de répéter deux fois le programme qu'on lui présente. CQFD.

Dans un même effort, et en suivant le même raisonnement, on peut alors définir l'addition – λmλnλfλxm(f)(n(f)(x)) –, la multiplication – λmλnλfm(n(f)) –, bref, toute l'arithmétique...

En achevant un travail commencé cinquante ans auparavant, Jean-Louis Krivine est finalement parvenu à démontrer, en 1997, que le lambda-calcul permet en réalité d'exprimer tous les raisonnements ainsi que toutes les structures mathématiques possibles. Non content d'être un langage informatique, ce monde fermé de formules qui s'appliquent les unes sur les autres dans un enchevêtrement extraordinairement complexe d'instructions est donc aussi un langage mathématique universel. Et si le lambda-calcul était ni plus ni moins le langage universel de la pensée ?

suite de la p. 41

théorèmes, qu'ils percent les belles vérités d'un monde idéal. Ils se leurrent ! Selon la théorie de Krivine – lui-même logicien… –, tout mathématicien est en fait un programmeur qui s'ignore. Car, passées à la moulinette du lambda-calcul, qui, rappelons-le, est un langage informatique, les démonstrations de leurs théorèmes peuvent dès lors être traduites… révélant des lignes de code informatique totalement insoupçonnées jusque-là ! (voir article, p. 50). Autrement dit, les mathématiciens ont le privilège de pouvoir changer de niveau, de monter jusqu'à des sommets d'abstraction en créant des langages et des structures très complexes, et de descendre vers les niveaux inférieurs en développant leurs calculs ; inconscients de leur propre rôle, ils essayent en fait de comprendre comment un système aussi simple que le lambda-calcul crée sous leur propre crâne des langages de programmation et des programmes informatiques aussi élaborés… Sans le savoir, lorsqu'ils démontrent des théorèmes, ils ne font que réécrire les programmes de leur propre cerveau ! « Mais ils ne font pas le boulot jusqu'au bout », ajoute Jean-Louis

L'intelligence dévoilée

Selon la théorie de Jean-Louis Krivine, plusieurs couches se superposent dans notre cerveau. En bas, ce sont les neurones, desquels émerge la couche du langage logique (le lambda-calcul), d'où émergent des langages de plus en plus évolués, jusqu'aux langues que nous parlons. De même, se superposent dans l'ordinateur des couches, depuis les électrons jusqu'aux langages de programmation, qui permettent aux informaticiens d'écrire leurs logiciels. Conséquence : à partir du lambda-calcul, véritable siège de l'intelligence, les processus cognitifs humains peuvent être mathématiquement traduits, révélant leurs "programmes".

Krivine. « Il faut le terminer en redonnant aux théorèmes leur vrai sens, en montrant à quels programmes ils correspondent, en montrant quelle tâche ils exécutent véritablement dans le cerveau. »

INFORMATICIEN AVANT LA LETTRE

Un seul exemple ici. On croyait que Kurt Gödel nous parlait des limites intrinsèques de la démarche mathématique. Faux. En élaborant ses concepts de logique, le génial mathématicien autrichien décryptait les mécanismes de sa propre pensée : il reconstituait peu à peu (et c'est là son génie) les couches de langage intermédiaires enfouies dans les plus profonds arcanes de son cerveau, entre la couche imprimée du lambda-calcul et celle, exprimée, de l'allemand, sa langue naturelle. Car, une fois traduit dans le langage du lambda-calcul, son célèbre théorème d'incomplétude se révèle ni plus ni moins un classique programme d'ordinateur, alors que l'informatique n'était même pas inventée. Un programme qui pourrait s'avérer une clé décisive pour pénétrer les profondeurs de nos rêves... (voir encadré, ci-dessous).

C'est ici un autre bouleversement qu'induit la théorie de Jean-Louis Krivine. Tandis que les psychanalystes essayent d'atteindre les couches du sujet par le haut, en partant du langage parlé, il devient pos-

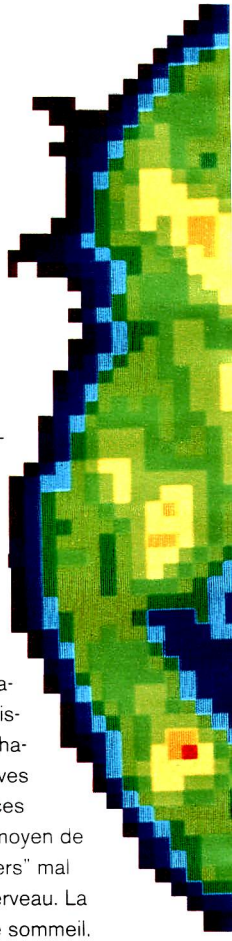
Théorème de Gödel
La vraie clef des songes ?

En 1931, le logicien autrichien Kurt Gödel démontre l'un des plus célèbres théorèmes : le théorème d'incomplétude, qui affirme qu'il existera toujours des vérités que les mathématiques ne pourront pas démontrer et dont le sens a donné le vertige à un nombre incalculable de penseurs.

Jean-Louis Krivine, lui, a simplement traduit dans le lambda-calcul la démonstration de ce théorème. Il tente, depuis, de démêler la longue ligne de code informatique sur laquelle il est tombé. « Je n'ai pas encore réussi à en comprendre vraiment la signification informatique, avoue-t-il, mais certains indices me font penser qu'il correspond au programme réparateur du système de fichiers. » Ce programme informatique, présent dans tous les ordinateurs, est utile lorsque la machine est éteinte brusquement, suite, par exemple, à une coupure de courant : lors du démarrage suivant, il fait défiler les fichiers et range ceux qui n'ont pas été correctement fermés, rendant pendant ce temps inutilisable l'ordinateur.

La situation est déjà troublante : Kurt Gödel aurait donc élaboré un programme informatique indispensable au bon fonctionnement d'un ordinateur de bureau, mais dont la tâche n'a, *a priori*, rien à voir avec son idée mathématique de départ – et tout cela, avant que l'ordinateur ne soit inventé ! Mais ce n'est pas tout : selon la théorie de Krivine, ce programme doit aussi être présent dans notre cerveau, le rendant inutilisable pendant qu'il tourne. Quel pourrait donc être ce processus cognitif très contraignant ? « Cela pourrait être une des fonctions du sommeil », propose le logicien, tout sourire. « Pendant que l'on dort, un tas de choses défilent dans notre cerveau en veille. L'idée est séduisante, non ? »

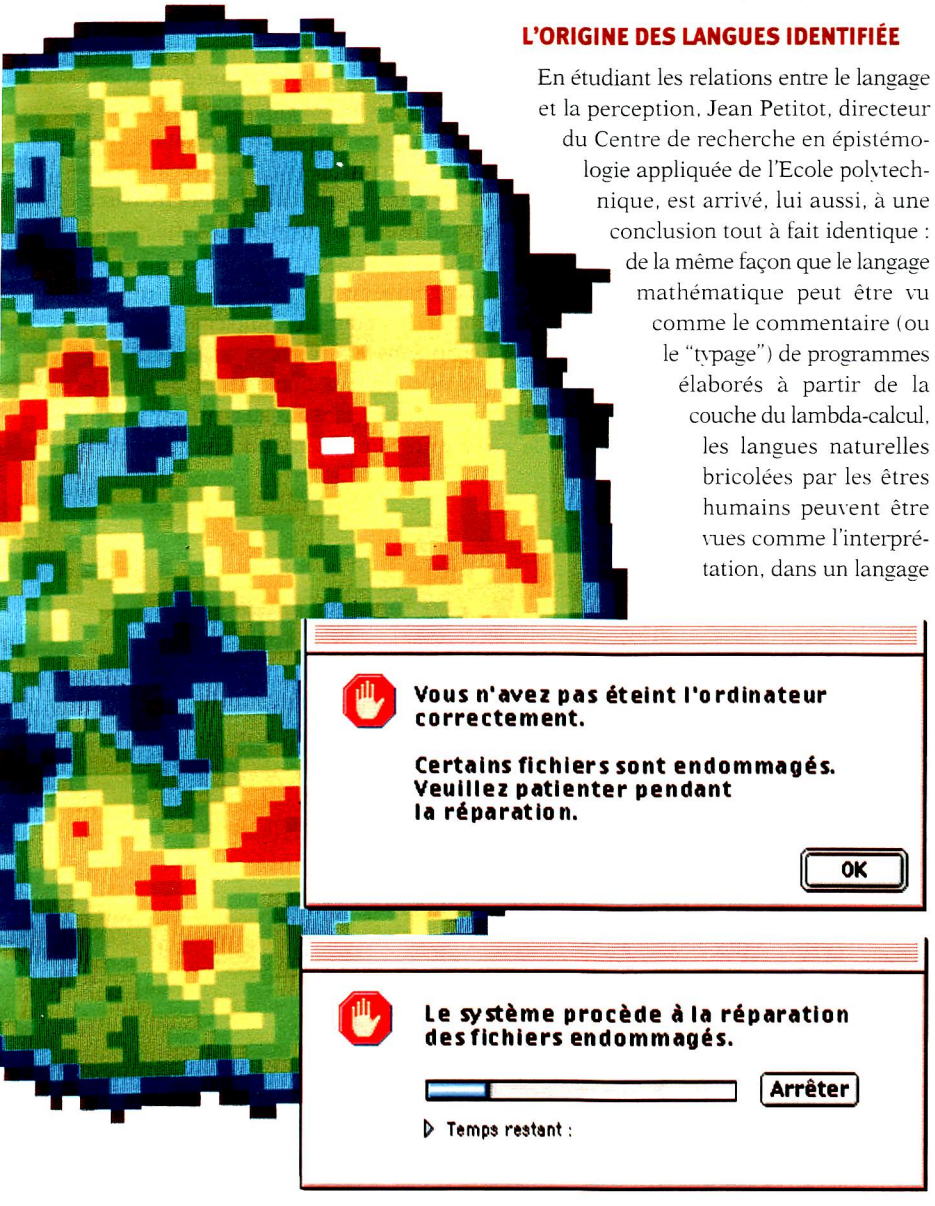
Vertigineuse, même. Le génial logicien autrichien croyait étudier les limites de la démarche mathématique, il pensait anéantir le programme de Hilbert prétendant atteindre toute vérité mathématique à partir de quelques axiomes et règles de raisonnement de départ. Il ne faisait, en fait, qu'étudier le fonctionnement de son propre cerveau. Il reconstruisait un programme informatique, peut-être indispensable à la psychanalyse ! Car nos rêves ne seraient, dans ces conditions, qu'un moyen de restaurer les "fichiers" mal fermés de notre cerveau. La nuit, pendant notre sommeil, on rangerait les programmes de la journée, on quitterait les applications mal fermées, et de vieux fichiers bugués ressurgiraient... toujours les mêmes.



sible de les aborder par le bas, en partant du lambda-calcul. De quoi reconstituer, peut-être, les couches de l'inconscient, du moi et du sur-moi qui structurent depuis un siècle l'approche psychanalytique...

Ce n'est pas tout. En linguistique, le lambda-calcul est déjà exploité pour structurer les langues naturelles parlées sur Terre. « L'idée n'est pas nouvelle », souligne Jean-Pierre Desclés, qui dirige le laboratoire Langage, logique, informatique et cognition (LaLIC), à l'université Paris-Sorbonne. « Dès les années cinquante, des logiciens et linguistes ont vu dans le lambda-calcul la structure logique sous-jacente aux langues naturelles. En étudiant la sémantique des phrases, comme les relations entre un nom propre, un verbe et un prédicat, tous les énoncés, tous les mots, même, peuvent être vus comme des lambda-termes, c'est-à-dire comme des programmes informatiques. » Pascal Boldini, qui travaille dans la même équipe que Jean-Pierre Desclés, est « persuadé que c'est la bonne voie. Cela permet de formaliser les langues naturelles de façon beaucoup plus concise que n'importe quelle autre théorie linguistique. »

S. MORGAN/SPL/COSMOS



L'ORIGINE DES LANGUES IDENTIFIÉE

En étudiant les relations entre le langage et la perception, Jean Petitot, directeur du Centre de recherche en épistémologie appliquée de l'Ecole polytechnique, est arrivé, lui aussi, à une conclusion tout à fait identique : de la même façon que le langage mathématique peut être vu comme le commentaire (ou le "typage") de programmes élaborés à partir de la couche du lambda-calcul, les langues naturelles bricolées par les êtres humains peuvent être vues comme l'interprétation, dans un langage

Une application neuronale

Et si le célébrissime théorème d'incomplétude de Gödel ne parlait pas de vérité mathématique ? Car traduit en lambda-calcul, ce théorème correspond à un programme informatique que l'on retrouve dans tous les ordinateurs : celui qui consiste à réparer les fichiers mal fermés. Quelle utilité un tel programme pourrait-il avoir au niveau neuronal ? Pour Jean-Louis Krivine, c'est évident : pendant notre sommeil, les rêves seraient l'application qui répare les fichiers mal fermés de notre cerveau. De quoi revoir toute l'approche psychanalytique...

de très haut niveau, des calculs neuronaux sous-jacents. L'existence de cette couche de lambda-calcul commune à tous les êtres humains expliquerait alors pourquoi deux langues en apparence aussi différentes que l'anglais et le mohawk (parlé par une tribu de la confédération iroquoise) présentent de si profondes analogies. Et pourquoi l'on a découvert un squelette commun à cinq cents langues et dialectes (voir *Science & Vie*, n° 990, p. 98). Jean Petitot en est convaincu : « La théorie de Krivine est la première thèse sérieuse sur l'origine du langage chez les hominidés. »

C'EST LE PC QUI MIME LE CERVEAU !

Cela dit, l'idée de comparer notre cerveau et l'ordinateur n'est pas nouvelle. Elle date même des débuts de l'informatique, puisque Alan Turing et John von Neumann conçurent le premier ordinateur, la fameuse machine Mach 1, comme une copie du cerveau humain. Mais cette analogie a pris du plomb dans l'aile. Aujourd'hui encore, les informaticiens sont totalement dépassés par la complexité des programmes à construire, cela rien que pour demander à une machine d'aller chercher du pain juste au coin de la rue... Avec la théorie de Krivine, cette comparaison retrouve pourtant ce qu'elle portait à l'origine d'intuition féconde. Mais rien de trivial ni de mécanique ici. Au contraire. « Ce n'est pas notre cerveau qui ressemble à l'ordinateur, explique le logicien, mais c'est l'ordinateur qui essaie à toute force, par programmeur interposé, de ressembler à notre cerveau. » Un retour à l'envoyeur, en quelque sorte. Là encore, le changement de point de vue est radical. Concrètement, cette ressemblance ne se situerait pas au niveau du *hardware*, de la machine proprement dite, faite de fils électriques ou de neurones, mais au niveau du *software*, du système de logiciels implémentés dans la mémoire. Si le lambda-calcul intervient entre la couche des circuits électroniques et les langages évolués de programmation dont se servent les informaticiens pour écrire leurs logiciels, c'est peut-être parce que notre cerveau procède de cette manière. A partir de là, toute activité calculatoire ou cognitive peut être vue comme l'expression de "programmes" implémentés dès le départ dans le cerveau.

Mais comment diable de tels programmes ont-ils pu être mis au point dans notre cerveau ? Simple : grâce à l'évolution dar-

LAUDATOR

La source de l'esprit

Tout se passe comme s'il n'y avait eu qu'une seule manière de penser depuis l'apparition de la vie sur Terre : des premiers cerveaux aux derniers ordinateurs, en passant par les premières conversations entre êtres humains et les toutes premières démonstrations mathématiques, toutes les pensées peuvent se résumer à un lambda-calcul sous-jacent, source unique de l'intelligence terrestre.

winienne ! Certes, il peut paraître surprenant que la sélection naturelle puisse parvenir à construire des lignes de code aussi complexes. Mais c'est ici sous-estimer la puissance de ce processus. Car, comme vient de le montrer Walter Fontana, la Nature est un programmeur hors pair.

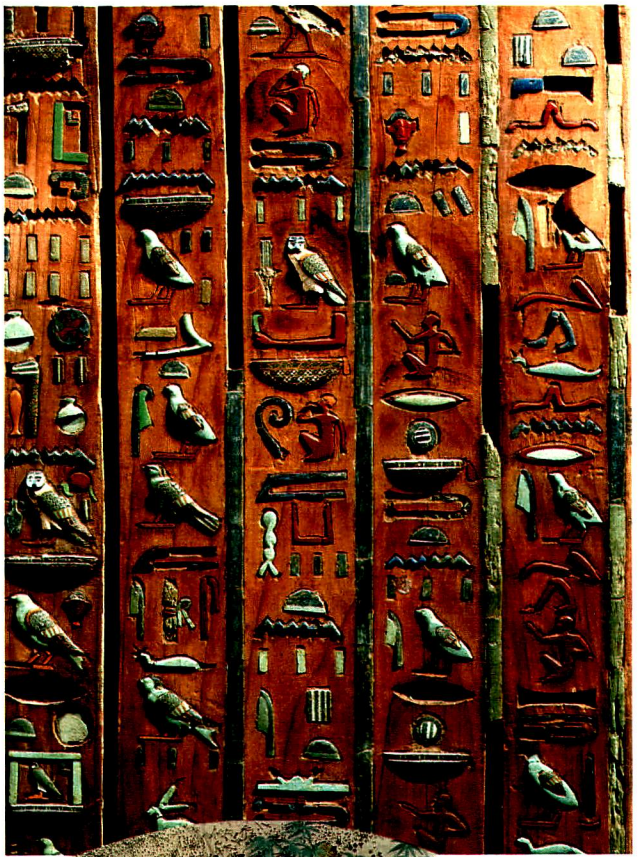
L'ALPHABET DE TOUTES LES PENSÉES

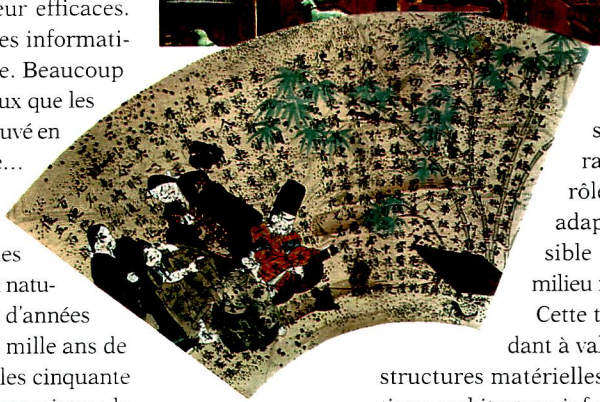
Ce chercheur autrichien de l'institut Santa Fe, aux Etats-Unis, s'est en effet inspiré de la sélection darwinienne pour construire un "programme du prédécesseur", soit une application qui répond 12 quand on lui donne le nombre 13, ou 6 pour 7. Or, en langage lambda-calcul, ce programme, en apparence très simple, s'est, étrangement, révélé très difficile à mettre au point. Idéal pour un test. Parti d'une séquence quelconque (un lambda-terme choisi au hasard), Walter Fontana l'a fait muter aléatoirement, en sélectionnant les programmes les plus proches de la tâche désirée, avant de lancer de nouvelles mutations. Résultat : au bout de plusieurs centaines de milliers de générations, il a obtenu quatre programmes du prédécesseur efficaces. Trois étaient déjà connus des informaticiens, mais pas le quatrième. Beaucoup plus court, simple et ingénieux que les autres, il n'avait jamais été trouvé en cinquante ans d'informatique...

Tout se passe donc comme s'il n'y avait eu qu'une seule manière de penser pendant les millions d'années de sélection naturelle, les dizaines de milliers d'années de babils humains, les trois mille ans de recherche mathématique et les cinquante ans de programmation informatique : le lambda-calcul, la source unique de l'intelligence, cet « alphabet des pensées humaines » qu'entrevoyait déjà le philosophe et mathématicien allemand Gottfried Wilhelm Leibniz au XVII[e] siècle, sans pouvoir aller plus loin. Du coup, l'adéquation des mathématiques avec la réalité, qui déconcerte depuis des lustres les épistémologues, ne serait plus si "déraisonnable". Si l'on admet les mathématiques comme étant le décodage de programmes écrits dans le cerveau, il n'y a rien d'étonnant à ce que ces programmes soient en profond accord avec l'environnement, sinon, ils rempliraient bien mal leur rôle, qui consiste à adapter le mieux possible l'organisme à son milieu naturel.

Cette théorie reste cependant à valider. Il faut que les structures matérielles de cette hypothétique architecture informatique soient effectivement observées dans notre cerveau. « On ne tombera peut-être pas exactement sur le lambda-calcul, pressent Jean Petitot, mais sur un "gamma-calcul" tout à fait analogue. Mais, une fois ce gamma-calcul connu, toutes les thèses de Krivine, selon moi, sont justes. »

Ce serait alors l'une des plus importantes découvertes scientifiques de tous les temps. En revenant aux sources de la pensée humaine, le logicien français promet aux chercheurs en intelligence artificielle, informaticiens, mathématiciens, linguistes,





Une seule langue

Rien de commun, a priori, entre les idéogrammes chinois, les hiéroglyphes égyptiens et un texte en français. Pourtant, des linguistes voient sous la plupart des langues parlées par l'homme une seule structure : celle d'un langage informatique.

G. DAGLI ORTI

psychologues, psychanalystes, neurologues et autres cognitivistes de se rendre compte qu'ils étudient tous le même objet, mais pas sous le même point de vue, pas au même niveau. Chacun pourra alors profiter du savoir accumulé dans les autres disciplines.

LA QUATRIÈME HUMILIATION ?

C'est sûr, cette théorie aura les plus importantes conséquences en informatique. Financièrement, en tout cas. Dépassés par la complexité des programmes à mettre au point, les informaticiens nagent en effet dans une mare de *bugs*. Pour de plus en plus de chercheurs, les lambda-correspondances entre mathématiques et informatique, maintenant solidement établies, sont le seul moyen d'éradiquer ces *bugs*. Et en prolongeant ces correspondances jusqu'à notre organe de pensée, Jean-Louis Krivine ouvre de très prometteuses nouvelles pistes de programmation (voir article, p. 54).

De même, ce que l'on sait du fonctionnement du cerveau devrait faire un bond en avant. Pour plonger dans ses arcanes, il suffit de décrypter ses programmes, en exploitant les connaissances accumulées par les mathématiciens et les informaticiens. « Il y a plus à comprendre sur le cerveau dans l'analyse d'un système d'exploitation que n'importe où ailleurs », prophétise Jean-Louis Krivine. Le fonctionnement parfois névrotique d'un PC permettra peut-être de comprendre certaines pathologies psychologiques…

Tant de bouleversements n'iront pas sans douleur. Car, après l'humiliation cosmologique de Galilée qui a chassé l'homme du centre de l'Univers, l'humiliation biologique de Darwin, qui l'a rejeté dans la même cour que les animaux, et l'humiliation psychologique de Freud, qui lui dénie la maîtrise consciente de ses actes, Jean-Louis Krivine nous inflige peut-être une quatrième humiliation, neurologique cette fois : il assimile notre cerveau aux simples circuits électriques d'un ordinateur de bureau… Nous pensions avoir des pensées ; nous ne faisons que reproduire des calculs qui moulinent depuis des millions d'années. Mais ces humiliations ne sont qu'apparentes. La dignité humaine n'est-elle pas, justement, de perdre ses illusions? ■

© KASPAROV/DEEPER BLUE

A qui perd gagne…

Le 11 mai 1997, le champion du monde des échecs Gary Kasparov perd contre l'ordinateur Deeper Blue. Une cruelle défaite pour l'idée que l'homme se fait de lui. Pourtant, il n'a été vaincu que par son intelligence retournée contre lui…

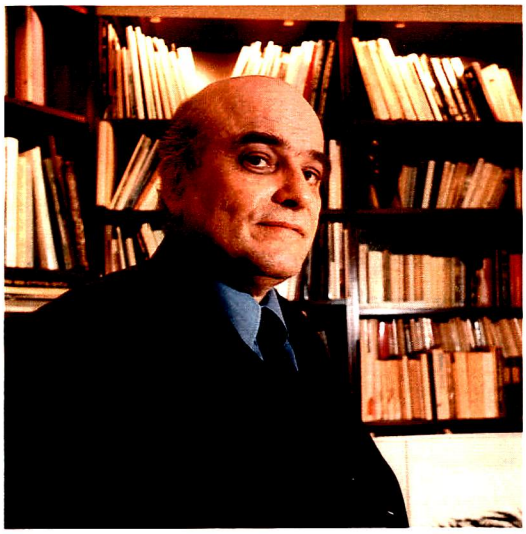
"Il s'agit

Jean Petitot
Cognitiviste, directeur du Centre de recherche en épistémologie appliquée de l'Ecole polytechnique.

Que pensez-vous du changement de point de vue radical que propose Jean-Louis Krivine?
Je pense que, sur les points essentiels, Krivine a raison. Certains théorèmes mathématiques peuvent sans doute se concevoir comme un typage, une sorte de commentaire, de programmes compilés dans le cerveau par l'évolution : il s'agirait du commentaire écrit dans un langage de haut niveau de ce programme écrit, lui, en langage de bas niveau. Certaines mathématiques seraient ainsi une sorte de "décompilation" du code neuronal. Il s'agit là d'une idée puissante.

Pourquoi cette théorie est-elle si convaincante?
C'est la meilleure réponse que je connaisse à un problème que posait Kant il y a bien longtemps : pourquoi des structures abstraites comme la géométrie et le calcul différentiel s'appliquent-elles *a priori* à l'expérience? D'après Krivine, c'est que nous n'avons accès aux phénomènes qu'à travers des calculs implémentés neuralement et qui, résultant de l'évolution, sont donc forcement adaptés à la réalité. Si la géométrie et le calcul différentiel reconstruisent certains de ces programmes, ceux-ci sont donc forcément adaptés à la réalité. Pas plus

d'une théorie très puissante"



P. LAFAY/EDITING POUR SCIENCE & VIE

que les autres animaux, nous n'avons conscience de ces calculs, mais nous avons les capacités réflexives permettant de trouver les structures mathématiques qui les typent. C'est un argument très fort pour expliquer la "miraculeuse" adéquation des mathématiques avec la réalité.

Pensez-vous que notre cerveau contienne de vrais programmes ?

C'est un problème délicat. De façon indépendante, en étudiant les relations entre le langage et la perception, je suis arrivé à des résultats assez analogues : l'énoncé d'un jugement perceptif, qui transforme une scène perçue en structure prédicative, peut être considéré comme le typage du calcul neuronal calculant cette scène. Imaginez un ballon rouge. Du côté de la perception, cette scène correspond d'abord à un flux optique pixelisé sur la rétine. Flux qui est ensuite transformé, *via* un ensemble d'algorithmes neuronaux (traitements de la couleur, de la texture…), en un objet géométrique constitué d'un domaine spatial, délimité par des bords et remplis par des qualités sensibles, ici, sa couleur. Mais, du côté du jugement, nous trouvons l'énoncé «le ballon est rouge». Pour comprendre comment nous passons de la perception à son énoncé, nous devons comprendre le changement de format qui fait passer de la structure topologico-géométrique de la scène visuelle à une structure logico-syntaxique faisant intervenir un sujet, un verbe… Avec certains collègues, j'ai essayé de le faire en utilisant des outils qui relient géométrie et logique : la théorie des faisceaux et celle des topoï, dont les applications à l'informatique sont déjà nombreuses. Selon moi, ces théories permettent de comprendre comment la géométrie visuelle peut servir de matériel syntaxique pour une grammaire d'énoncés descriptifs. On retrouve l'idée de typage à la Krivine. Mais il faudrait sans doute généraliser sa théorie : ce seraient les concepts et les jugements en général, et pas seulement les théorèmes mathématiques, qui seraient des typages linguistiques de calculs cognitifs de bas niveau. Il s'agit là de l'une des premières thèses formelles sérieuses sur l'origine du langage.

Le lambda-calcul est-il alors «l'alphabet des pensées» dont ont parlé des philosophes ?

Un certain nombre de psychologues cognitivistes, comme Jerry Fodor, font depuis longtemps l'hypothèse qu'il existe un langage de la pensée assez proche des langages logiques de programmation. Est-ce le lambda-calcul ? C'est une question expérimentale qui est loin d'être évidente. Selon moi, le langage machine des calculs neuronaux n'est pas le lambda-calcul. Ce qui compte est ce que les neurobiologistes nomment l'architecture fonctionnelle des aires cérébrales, c'est-à-dire, en quelque sorte, le *hardware* neuronal. Or, il est très différent d'une machine de Turing. Il n'y a pas de zones mémoires, d'adresses, de pointeurs, de piles, etc., qui permettraient d'implémenter un lambda-calcul. Mais, à ceci près, je crois que les thèses de Krivine à propos du passage du langage machine aux langages de haut niveau sont essentiellement justes.

Comment approfondir ces hypothèses ?

Le développement de ces recherches exige une interdisciplinarité réelle, j'aime à dire "intrinsèque", car imposée par l'objet d'étude lui-même. Il nécessiterait donc des équipes constituées de spécialistes de la logique, de l'informatique, de la géométrie différentielle, des neurosciences intégratives et cognitives, de la linguistique… C'est un champ de recherche dont les enjeux scientifiques et technologiques sont extraordinaires. Mais l'expérience montre que le système français est mal adapté à la promotion de tels travaux. Les rigidités des disciplines établies les confinent trop dans les marges. Nous demeurons encore loin de ce qui s'est massivement développé dans les autres pays. Mais les enjeux de l'approche scientifique rigoureuse des processus mentaux sont tels que leur progrès est irrésistible.

Au-delà des mathématiques

Unifier les maths, l'informatique et le cerveau, telle est la formidable prouesse accomplie, après un siècle de recherches en logique, par Jean-Louis Krivine. Pour ce logicien de réputation mondiale, "l'alphabet des pensées humaines" n'est plus un rêve. Explications.

Jean-Louis n'est pas le plus célèbre des Krivine. On connaît mieux ses cousins : Alain, porte-parole de la Ligue communiste révolutionnaire, qui s'est présenté plusieurs fois à la présidence de la République, et Emmanuel, un grand musicien qui dirige actuellement l'Orchestre français des jeunes, à Paris. Jean-Louis sera peut-être pourtant un jour le plus connu de tous : mathématicien, la théorie qu'il a élaboré révolutionne déjà notre vision du cerveau.

Mais comment un mathématicien peut-il nous parler du cerveau? Comment peut-on déceler, dans les formules alambiquées qui s'étalent sur un tableau noir, la trace des plus profondes pensées humaines? Tranquillement assis à son bureau, au cinquième étage d'une tour moderne et anonyme, dans le laboratoire d'informatique théorique de l'université de Jussieu, Jean-Louis Krivine fait semblant d'être étonné par la question : « Le cerveau a toujours été mon seul sujet de recherche, mais la logique est un outil irremplaçable pour explorer sa structure profonde. » Ni modeste, ni vantard, il croit aujourd'hui être arrivé au bout de sa logique, au bout de son intuition. La voix calme et le regard pétillant, il explique ce chemin paradoxal qui permet à un travail de logique mathématique d'analyser le fonctionnement de l'esprit même qui l'a produit.

« Je me suis intéressé au cerveau dès que j'ai commencé à faire des mathématiques, vers l'âge de 15 ans. Mais mes recherches n'ont vraiment commencé qu'au début des années quatre-vingt, lorsque j'ai entendu parler de la correspondance de Curry-Howard. J'ai immédiatement abandonné le domaine de l'analyse fonctionnelle pour me consacrer entièrement à l'étude de cette correspondance. C'était cela que j'avais toujours voulu étudier. »

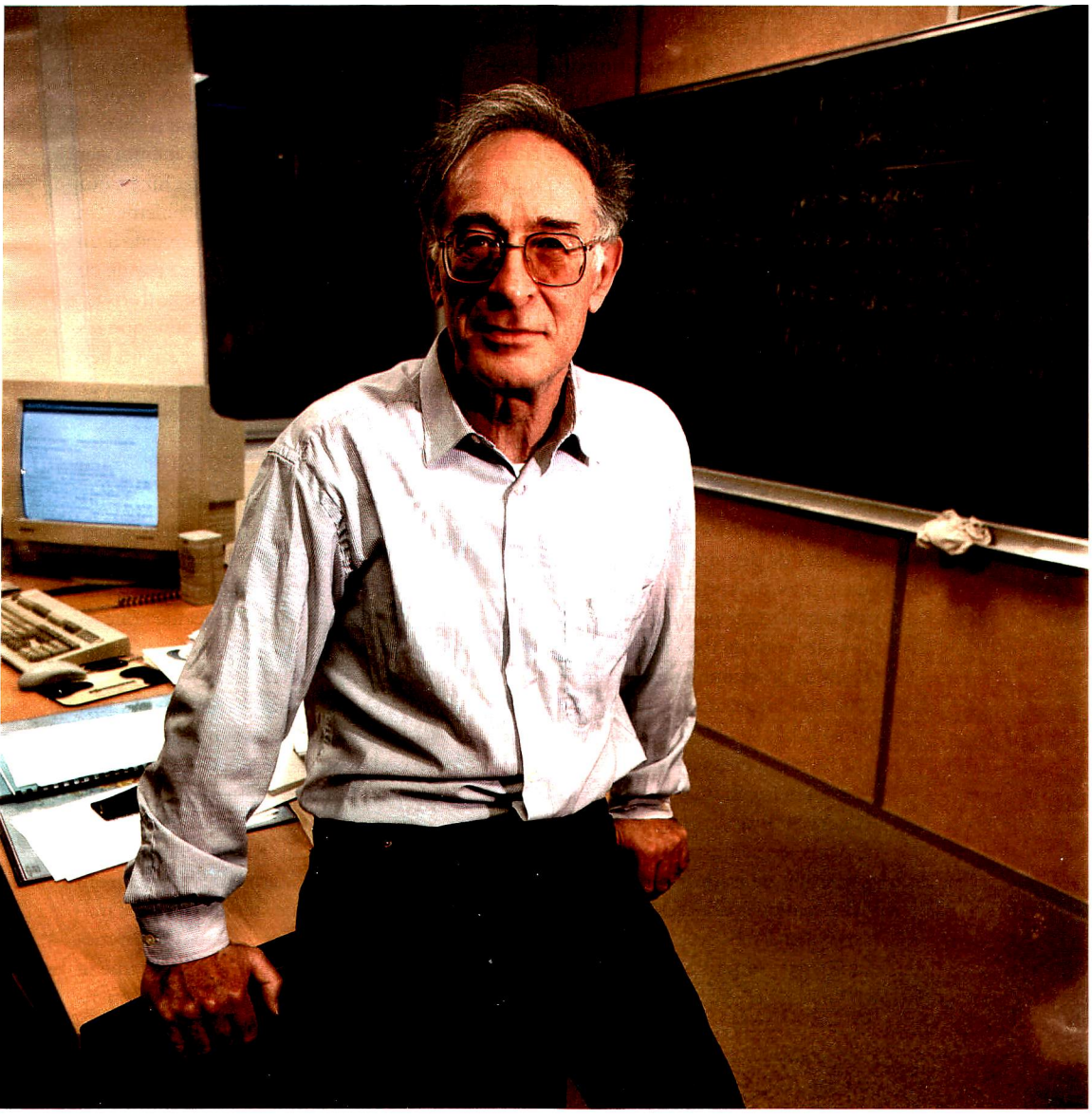
L'IMPROBABLE CORRESPONDANCE

Si vous n'êtes pas chercheur de pointe en informatique théorique ou en logique mathématique, vous n'avez probablement jamais entendu parler de la correspondance de Curry-Howard, clé de voûte de la théorie de Krivine. Issue d'une remarque faite en 1958 par le logicien américain Haskell Curry, explicitée onze ans plus tard par William Howard, elle affirme que certaines démonstrations mathématiques correspondent à des programmes informatiques.

Rien d'étonnant *a priori*. Une démonstration est en effet achevée si, des hypothèses à la conclusion, on peut l'expliquer à un parfait imbécile, tout juste capable d'appliquer les quelques règles élémentaires de raisonnement qu'on lui a inculquées... Et l'ordinateur est le plus bel imbécile que l'on connaisse! Toute démonstration est donc mécanisable. C'est même un théorème mathématique : le théorème de complétude, démontré en 1930 par le logicien autrichien Kurt Gödel (1).

Cette correspondance ébauche cependant

> **« Le cerveau a toujours été mon seul sujet de recherche. Depuis l'âge de 15 ans, je n'ai pas cessé de l'étudier. »**
>
> **Jean-Louis Krivine,** mathématicien.



un véritable dictionnaire mathématico-informatique. Elle montre que, malgré leurs langages apparemment fort différents, ces deux constructions du génie humain peuvent s'exprimer dans une seule et même langue primordiale : le lambda-calcul.

Ce langage, créé en 1932 par le mathématicien américain Alonzo Church, ne comprend que deux règles grammaticales. Il est très simple à définir, mais très difficile à maîtriser (voir l'encadré p. 42).

(1) A ne pas confondre avec le théorème d'incomplétude, traduit par Jean-Louis Krivine (voir encadré, p. 44).

Pourtant, Church a montré qu'il est suffisamment riche pour traduire les principaux objets mathématiques. Le lambda-calcul serait cependant resté confidentiel si les informaticiens ne s'étaient pas rendus compte, dans les années cinquante, que c'est un parfait langage de programmation : tout lambda-terme peut être vu comme un programme informatique.

Mais le génie d'Haskell Curry est d'avoir remarqué que ce dialecte ésotérique permet aussi de traduire certaines règles de déduction utilisées en mathématiques, comme l'implication. Or, une démonstra-

tion mathématique n'est qu'une succession de règles de déduction qui s'appliquent sur des objets. Les démonstrations peuvent donc être traduites, ligne après ligne, dans le langage du lambda-calcul, pour devenir de véritables programmes informatiques. « C'est un travail mécanique », explique Jean-Louis Krivine. « A chaque pas de la démonstration, on assemble les pièces détachées du lambda-calcul qui, à la fin, constituent le programme recherché. »

La correspondance entre théorème et programme devient ainsi très élégante : l'énoncé du théorème correspond à la tâche que le programme accomplit lorsqu'on l'exécute, le système d'axiomes sur lequel se fonde la démonstration correspond au système d'exploitation qui fait tourner le programme (Windows, Mac OS, UNIX...), et les lignes de démonstration du théorème, aux lignes de code du programme (de même qu'il existe plusieurs démonstrations possibles pour un théorème, il y a plusieurs façons de programmer une même tâche).

La seule marque d'ordinateur sur le marché depuis plusieurs centaines de milliers d'années, c'est le cerveau humain.

Pendant longtemps pourtant, ce dictionnaire n'a pas semblé bien passionnant. Depuis les premières pages écrites par Curry, de grands logiciens, comme le Français Jean-Yves Girard, l'ont peu à peu complété. Mais les seules démonstrations qu'il permettait de traduire étaient les calculs mathématiques dont les programmes correspondants consistaient à exécuter ce calcul. Rien de révolutionnaire là-dedans...

UN RAISONNEMENT REBELLE

A l'époque, le dictionnaire était en fait incomplet. Un raisonnement mathématique échappait à toute traduction : le raisonnement par l'absurde, qui prêche le faux pour savoir le vrai, qui suppose le contraire de ce que l'on veut prouver, afin d'arriver à une contradiction. Ce raisonnement, couramment utilisé et terriblement efficace, permet d'étudier des objets sans que ceux-ci aient été construits au préalable. Il permet donc aux démonstrations d'être autre chose que de simples calculs.

« Longtemps on a cru que seules les démonstrations qui n'utilisent pas ce raisonnement pouvaient être traduites dans le lambda-calcul », se souvient Jean-Louis Krivine. « Mais, en 1990, les informaticiens américains Matthias Felleisen et Timothy Griffin ont découvert que le raisonnement par l'absurde correspond, en fait, à un petit programme présent dans tous les ordinateurs : les instructions d'échappement. » Ces instructions consistent à mémoriser un état de référence de la machine pour pouvoir y revenir en cas de problème. On les voit en action lorsqu'on demande à un ordinateur de lire une disquette absente du lecteur et qu'il affiche un message d'erreur. « Avouez qu'il n'était pas évident que les instructions d'échappement aient un lien avec le raisonnement par l'absurde... » Ce n'est que la première des surprises.

MI HERCULE, MI CHAMPOLLION

Maintenant que tous les raisonnements sont traduits dans le lambda-calcul, le dictionnaire est devenu universel (2). Tout théorème peut être lu en termes informatiques. Jean-Louis Krivine décide donc de se lancer dans la traduction de quelques fameux théorèmes. Avant cela, il doit choisir un système d'axiomes sur lesquels fonder les démonstrations à traduire (de la même façon que, avant de se lancer dans l'écriture de lignes de code, il faut savoir sous quel système d'exploitation tournera le programme). Jean-Louis Krivine choisit le système d'axiomes de Zermelo-Fraenkel, « qui a été adopté comme standard par les mathématiciens pour ses qualités de commodité et d'élégance ». En 1997, il donne la traduction des six axiomes qui le constituent. L'exégèse peut commencer.

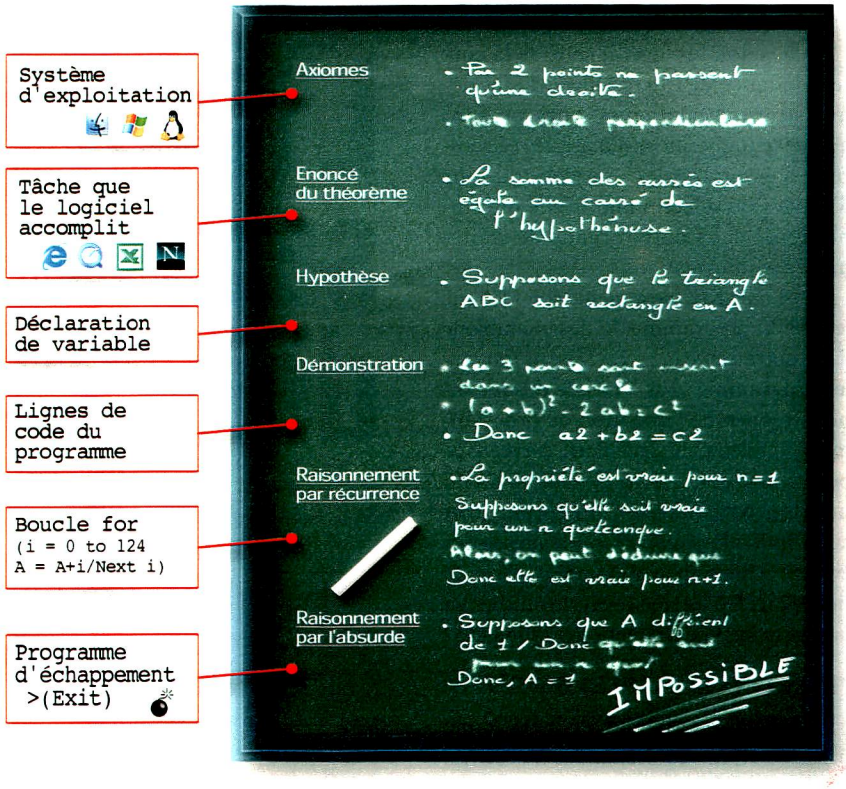
Le logicien choisit un texte particulier, « le plus important de tous », selon lui : le théorème de complétude de Gödel, prouvé en 1930 par le génial logicien autrichien, qui affirme, comme on l'a vu, que les mathématiques sont mécanisables. Armé de son dictionnaire, il traduit sa démonstration et cherche le sens du programme.

La tâche est herculéenne : les démonstrations qui exploitent le raisonnement par l'absurde donnent un lambda-terme très emberlificoté ; et il faut une parfaite agilité logique, une bonne connaissance informatique, du flair, de l'astuce et

(2) Il faut ajouter au lexique du lambda-calcul le terme (cc), qui correspond à l'instruction d'échappement, équivalent informatique du raisonnement par l'absurde.

Démontrer, c'est programmer

Un nouveau "dictionnaire" est en train de s'élaborer. Surprenant et inattendu, il traduit, grâce à la correspondance de Curry-Howard, en termes de programmation informatique toutes les notions utilisées dans les démonstrations mathématiques. Le système d'axiomes correspond au système d'exploitation, les théorèmes aux programmes, les types de raisonnement à des types d'instructions, etc. Ce dictionnaire montre, *in fine*, que démontrer, c'est programmer.



même un peu de chance pour voir à quoi ce programme peut bien servir. « Nous sommes un peu comme les égyptologues avant Champollion : nous savons que ces hiéroglyphes veulent dire quelque chose, mais nous ne savons pas quoi. »

Mi-Hercule, mi-Champollion, Jean-Louis Krivine réussit finalement à entrevoir le sens caché du théorème de complétude de Gödel – un sens inattendu : « Ce théorème semblait correspondre à un logiciel bien connu des informaticiens. Appelé désassembleur, il fait tourner instruction par instruction d'autres programmes afin d'en comprendre le code, de le modifier ou de trouver les erreurs de programmation. »

Un détail cependant le chiffonne : « La traduction de la démonstration du théorème ne correspondait pas exactement à un désassembleur. Plus précisément, le programme que j'avais écrit stipulait de faire du "pas à pas" partout, sauf lorsqu'il rencontre les instructions d'échappement, instructions qu'il doit systématiquement sauter. Je ne savais qu'en penser : pourquoi mon programme évitait-il donc ces instructions ? »

C'est en discutant avec Emmanuel Saint-James, un collègue informaticien spécialiste du langage LISP, que Jean-Louis Krivine trouve la réponse. Déconcertante. « Il m'a dit ce que tous les spécialistes de la programmation savent : un désassembleur doit nécessairement inclure cette particularité. On ne peut pas demander au désassembleur de faire du pas à pas sur les instructions d'échappement, sinon, il plante l'ordinateur. »

LA VÉRITÉ ENFOUIE DES MATHS

La situation est troublante. Sans le savoir, en traduisant simplement un vieux théorème de logique démontré dans les années trente, alors qu'il n'existait pas l'ombre d'un ordinateur, Jean-Louis Krivine a réécrit de façon très précise un programme informatique très élaboré qui intègre de façon parfaite la sécurité d'un système informatique... Mais à quel ordinateur Kurt Gödel pouvait-il donc inconsciemment se référer ? Pour Jean-Louis Krivine, il n'y a qu'une réponse possible : « Il s'agit de la seule marque d'ordinateur commercialisée à l'époque, et qui est d'ailleurs sur le marché depuis plusieurs centaines de milliers d'années : le cerveau humain. »

LAUDIATOR

L'informatiqu sauvée des bugs

Voilà, selon lui, le vrai sens de la correspondance de Curry-Howard : les démonstrations mathématiques ne sont qu'une tentative inconsciente de réécriture des programmes que des millions d'années de sélection naturelle ont implémenté dans notre organe de traitement des informations.

Jean-Louis Krivine a même son idée sur le rôle du programme du désassembleur dans notre cerveau : « En faisant tourner les programmes pas à pas, le désassembleur recherche l'intention du programmeur. Implémenté dans notre cerveau, ce programme donne donc la possibilité à un individu de connaître l'intention d'un autre individu. C'est un avantage sélectif évident. » Sans en avoir jamais eu l'intention, Kurt Gödel ne faisait qu'étudier sa propre recherche d'intention. Vertigineux !

"FAIRE ENTRER DANS LES MŒURS"

Le véritable dictionnaire est donc trilingue : il fait correspondre théorèmes mathématiques, programmes informatiques... et activités cognitives. Jean-Louis Krivine le complète peu à peu. Mais le travail est gigantesque. C'est tout le corpus mathématique qu'il faut traduire. « Le problème n'est pas encore entré dans les mœurs », déplore le logicien. « Peu de mathématiciens voient la véritable essence de la correspondance de Curry-Howard. »

Publiée il y a quelques années par la revue *Bulletin of Symbolic Logic*, la correspondance entre le théorème de complétude et le désassembleur n'a pas encore été assimilée. « Le logicien qui a relu cet article avant publication était visiblement très compétent, explique Jean-Louis Krivine, mais il n'avait rien compris à mon travail : il me conseillait de supprimer la partie la plus importante, celle où j'explique le lien avec le désassembleur ! Je n'ai, bien sûr, tenu aucun compte de cette demande. Je crois que cet article n'a été finalement accepté que parce que j'y démontre incidemment le théorème de Gödel sans utiliser le raisonnement par l'absurde... Mais, pour moi, cette prouesse n'a aucune importance ! La seule chose qui m'intéresse, c'est le lien avec le désassembleur, le lien avec le cerveau. » ■

Devenus trop complexes, les programmes informatiques ne peuvent plus aujourd'hui être garantis sans bugs. La solution ? Les rendre aussi indubitables que des théorèmes, grâce au lambda-calcul. Une alliance entre les maths et l'informatique qui porte déjà ses fruits.

Les informaticiens ont un problème : plus leurs programmes sont sophistiqués, moins ils sont capables de savoir s'ils fonctionnent correctement. Or, ces programmes deviennent indispensables dans les secteurs les plus sensibles de l'économie et de la vie sociale. Et autant les bugs récurrents du logiciel Windows de Microsoft n'ont pas trop de conséquences (il suffit de redémarrer l'ordinateur), autant la catastrophe menace quand il s'agit d'un programme sur lequel repose le bon fonctionnement d'un avion, d'un métro, d'un outil chirurgical, d'une carte à puce ou d'un système de défense antimissile.

« La prise de conscience de ce problème est récente, estime Roberto di Cosmo, de l'Institut national de recherche en informatique et en automatique (INRIA). Il a fallu de retentissantes défaillances informatiques, avec des pertes financières gigantesques, comme le blocage de l'aéroport de Denver au début des années 90 à cause d'une défaillance dans le système de gestion des bagages, ou l'explosion d'Ariane 5 en 1996, à cause d'un bug dans un petit programme périphérique, pour que les ingénieurs, les industriels et les politiques prennent enfin cette question au sérieux. « Jusque-là, un logiciel était en effet considéré comme correct... tant qu'il ne s'était pas révélé défaillant. La commission d'enquête sur l'accident d'Ariane propose d'adopter maintenant le principe opposé, selon lequel un logiciel est présumé défaillant tant qu'il n'a pas été jugé correct. » Un changement de point de vue radical, mais nécessaire. « Les programmes informatiques actuels sont probablement les objets les plus complexes jamais construits par l'homme, reprend Roberto di Cosmo. Or, ils sont élaborés selon des méthodes empiriques, un bricolage encore acceptable il y a vingt ans qui ne l'est plus aujourd'hui. »

Petit bug, gros crash

L'explosion d'Ariane 5, le 4 juin 1996. Une erreur de programmation qui a fait voler en éclats les certitudes des informaticiens.

P. HERZOG/AFP FOR ESA/CNES

L'ampleur du problème fait frémir le monde de l'informatique. « Le niveau de complexité atteint par l'informatique a engendré une crise qu'il n'est plus possible de surmonter sans un immense travail collectif de recherche », avoue Paul Horn, responsable des laboratoires de recherche d'IBM. Intel, le constructeur de puces informatiques, a investi plusieurs millions de dollars dans cette chasse aux bugs. Et, dans les deux ans qui viennent, le gouvernement français prévoit un recrutement exceptionnel de chercheurs à l'INRIA et au CNRS, dans le tout nouveau département STIC (Science et technologie de l'information et de la communication).

COMMENT "PROUVER" UN LOGICIEL ?

« L'informatique moderne est confrontée à une crise des fondements, estime Pierre-Louis Curien, directeur du laboratoire "Preuves, programmes et systèmes" du CNRS et de l'université Paris-7. Malgré son développement effréné, c'est une science très jeune, encore largement à la recherche de ses concepts fondamentaux et unificateurs. » Mais comment prouver qu'un programme ne comporte pas d'erreur ? On peut bien sûr faire quelques tests, vérifier que le programme réagit correctement dans différentes situations. C'est ce que fait, par exemple, Microsoft pour débugger son logiciel d'exploitation Windows. Mais, face à des situations infi-

niment variées, ces tests ne peuvent être exhaustifs. Il est vain d'espérer prouver ainsi qu'un programme informatique un tant soit peu compliqué est correct, de la même façon qu'il est vain d'espérer prouver un résultat mathématique en le vérifiant sur quelques valeurs particulières. Non, ce qu'il faut, c'est un vrai raisonnement logique, une vraie démonstration, aussi rigoureuse que celle d'un théorème.

Et justement, les logiciens ont découvert un moyen de modéliser mathématiquement de nombreux paradigmes de la programmation. Ils ont construit un véritable traducteur qui transpose les lignes de codes informatiques en raisonnements logiques mathématiques. C'est la correspondance de Curry-Howard (voir article précédent). Sauvera-t-elle l'informatique?

P. DURAND/CORBIS SYGMA

Automate sécurisé

Les voitures de la ligne n° 14 du métro parisien sont complètement automatisées. Pour des raisons évidentes de sécurité, la RATP a dû prouver que les programmes informatiques qui gèrent ces déplacements ne contiennent aucun bug susceptible de provoquer un accident.

L'INSPECTEUR COQ EN ARBITRE

Cela fait vingt ans que les chercheurs fondent dessus leurs espoirs de rendre un jour les programmes aussi sûrs que les théorèmes. Durant les années 80, des informaticiens de l'INRIA ont ainsi mis au point le logiciel Coq, inspiré de cette correspondance. «C'est une sorte d'inspecteur automatique qui vérifie que les programmes sont corrects, explique Gilles Dowek, responsable de l'équipe chargée de développer le logiciel. Ecrire un programme en Coq demande un travail un peu plus important que de l'écrire dans un langage traditionnel, mais une fois qu'il est écrit, on est certain de son bon fonctionnement.»

Aujourd'hui, Coq en est à sa septième version et commence à devenir un éradiqueur de bugs efficace. Il est par exemple utilisé par Trusted Logic, une jeune entreprise installée à Versailles, pour garantir que les opérations effectuées par nos cartes à puce ne vont pas, sans raison, diviser ou multiplier par dix notre compte bancaire ou notre crédit téléphonique. «Coq formalise les programmes informatiques embarqués sur les cartes à puce afin de prouver la validité des parties logiques les plus complexes et les plus sensibles, précise Eduardo Gimenez, chef de projet dans la start-up. Cela permet aux constructeurs de démontrer les propriétés de sécurité nécessaires à l'obtention de certificats.» La Direction centrale de sécurité des systèmes d'information (DCSSI), sous l'autorité du Premier ministre, exige en effet «que toutes les caractéristiques de sécurité annoncées par les constructeurs de cartes à puce soient vérifiées.»

PROGRAMMER PAR L'ABSURDE

L'intérêt de Coq n'est pas seulement de fournir une garantie anti-bug. Il aide aussi les informaticiens à mieux comprendre la structure de leurs logiciels. Les fabricants de cartes à puce avaient par exemple un problème avec la taille du programme informatique vérifiant la validité des opérations effectuées sur leurs cartes. «Ce filtre occupait plusieurs méga-octets de mémoire et était réputé impossible à intégrer dans les cartes, se souvient Claire Loiseaux, responsable des évaluations sécuritaires et des méthodes formelles à Trusted Logic. En formalisant à l'aide de Coq les règles nécessaires à ce filtre, nous avons finalement réussi à le réduire à 8 kg-octets. La première licence de ce filtre a été acquise par Schlumberger et sera disponible dans la prochaine génération de cartes à puce proposée par le fabricant.»

Mais «les informaticiens sont loin d'avoir exploité tout le potentiel de la correspondance de Curry-Howard», estime Pierre-Louis Curien. Coq se limite en effet à traduire les programmes informatiques dans la logique constructiviste, c'est-à-dire en s'interdisant d'utiliser le raisonnement par l'absurde. Or, comme le souligne Michel Parigot, chercheur dans le laboratoire de Paris-7, «l'extension de la correspon-

dance au raisonnement par l'absurde au début des années 90 est une étape cruciale pour la formalisation de l'informatique. Elle promet de mettre au point des langages de programmation beaucoup plus naturels, fiables et lisibles.»

Le but n'est plus de prouver qu'un programme ne comporte pas de bug, mais de reconstruire le programme à l'aide d'un langage de programmation beaucoup plus naturel : le langage mathématique, patiné par trois mille ans d'expérience. La correspondance de Curry-Howard est ici utilisée dans l'autre sens : on part de raisonnements mathématiques que l'on traduit en programme informatique. Les informaticiens pourraient ainsi être aussi à l'aise avec leurs lignes de code que les mathématiciens le sont avec leurs lignes de démonstrations... C'est la voie qu'a choisie Jean-Louis Krivine, membre éminent du laboratoire parisien. En s'inspirant des principaux systèmes d'axiomes mathématiques, il pense pouvoir fonder de nouveaux systèmes d'exploitation plus fiables. Et en traduisant les théorèmes, il espère tomber sur des merveilles de programmes garantis sans bug.

VERS DES LOGIQUES EXOTIQUES

Ses premiers résultats sont plus qu'encourageants. En traduisant un des plus simples théorèmes qui existe, le logicien français est tombé sur un petit programme «d'une intelligence incroyable» qui pourrait révolutionner la façon de programmer les gros logiciels (voir encadré ci-contre). De plus, la théorie de Krivine qui prolonge ces correspondances jusqu'au cerveau ouvre de nouvelles pistes de programmation. Jean-Pierre Desclés, qui travaille à l'Institut des sciences humaines avancées, à Paris, propose, par exemple, de s'inspirer des syntaxes des langues naturelles pour mettre au point de nouveaux langages informatiques et faire ainsi profiter à l'informatique de plusieurs millénaires d'expérience de l'humanité dans le maniement de la langue.

Les recherches pour restructurer les fondements de l'informatique n'en sont qu'à leur début. D'autres logiques mathématiques plus exotiques sont explorées pour modéliser les calculs parallèles ou la gestion des données. Le responsable des laboratoires de recherche d'IBM propose, lui, de s'inspirer des "calculs autonomes" de notre corps, capable de réguler automatiquement les battements de cœur, la respiration ou la digestion en fonction des conditions externes. Ce dernier projet semble pour l'instant bien vague, mais il est significatif d'un profond changement de mentalité : si les informaticiens veulent franchir le mur de la complexité, ils vont devoir rompre leur isolement et s'inspirer des expériences accumulées par les mathématiques, la linguistique, voire la biologie. S'ils veulent s'en sortir, ils n'ont de toute façon pas le choix. ■

Programmation-objet
La logique du buveur

■ La programmation-objet est un des concepts clés de l'informatique. En faisant appel à des objets, des sous-programmes définis dans une bibliothèque de données, elle permet de concevoir de gros logiciels sans se perdre dans un océan de lignes de code. Les fondements de cette méthode de programmation vont peut-être bientôt être bouleversés. Jean-Louis Krivine s'est en effet attaqué au théorème du buveur, «le plus trivial des théorèmes non triviaux», selon lui. Ce théorème, qui n'a aucun intérêt mathématique, s'énonce ainsi : "dans tous les bars, il y a toujours une personne dont on peut dire avec raison que, si elle boit, alors tous les autres clients boivent aussi". Sa démonstration est simple. Prenons un bar au hasard. Soit tous les clients sont en train de boire et la phrase du théorème est vraie, quel que soit le client choisi. Soit il existe des clients qui ne boivent pas et il suffit de choisir l'un d'eux pour que la phrase "s'il boit, alors tout le monde boit" soit vraie, puisqu'en l'occurrence, la personne concernée ne boit pas... La traduction de cette démonstration dans le lambda-calcul est très courte : λz(cc)λk(z)λd(cc)λh(k)(z)h. Jean-Louis Krivine s'est pourtant cassé les dents sur ce petit programme pendant cinq ans. «Il construit plein de nouveaux programmes dont je n'arrivais pas à comprendre le sens.» Il ne l'a saisi que très récemment : «Les programmes qu'il fabrique ne doivent en fait pas être exécutés : ce ne sont que des noms, des noms qui permettent de baptiser les objets de façon précise et impeccable. Ce petit programme va donc peut-être permettre de restructurer toute la programmation-objet.» Le plus simple des théorèmes pourrait ainsi se révéler d'une importance informatique considérable !