Mathématique des programmes et programme des mathématiques

Colloque "La seconde vie de la logique mathématique"
E.N.S. 6 Avril 1992

Jean-Louis Krivine

Equipe de Logique mathématique, Université Paris VII, C.N.R.S.

e-mail krivine@logique.jussieu.fr

Qu'est-ce que les mathématiques, et pourquoi en fait-on? Voilà bien une question philosophique fondamentale à laquelle tout mathématicien doit faire face un jour ou l'autre. Or il se trouve que, dans mon domaine de recherches, à la frontière entre la logique mathématique et l'informatique, des outils puissants ont été découverts, grâce auxquels il me semble possible d'apporter une réponse à ce problème. C'est du résultat de ces réflexions que je vais vous parler ici. Pour attaquer cette question, j'emploierai la méthode habituelle du scientifique : se laisser d'abord guider par l'intuition sans trop se poser de questions, c'est-à-dire affirmer les choses sans les démontrer vraiment, simplement parce qu'elles s'imposent avec assez de force ; puis en tirer des conséquences, et voir si ça marche. Comme on dit familièrement, ça passe ou ça casse!

Je partirai de l'idée ou plutôt de la constatation suivante : quand on fait de la recherche mathématique, on a toujours l'impression d'explorer une réalité préexistante, et non pas de créer quelque chose ex nihilo. Prenons un exemple tiré de la théorie des nombres, le célèbre théorème des nombres premiers. Supposons que je cherche un équivalent du $n$ième nombre premier. Je vais commencer par faire des expériences, calculer le 100ième nombre premier (541), puis le 1000ième (7919), le 2000ième (17389), le 10000ième (104729) si j'ai beaucoup de courage. Ensuite, j'examinerai les résultats, et mon intuition, ou mon expérience me suggèrera la formule $n\log(n)$. Je voudrai alors m'en assurer, c'est-à-dire démontrer cette formule, soit par mes propres moyens, soit en cherchant dans la littérature. Au bout du compte, l'équivalent $n\log(n)$ se révèlera exact. Y a-t-il ici la moindre différence avec la démarche du physicien qui cherche une loi naturelle? Non : tout au long de ce travail, jamais le moindre doute ne m'aura effleuré sur la réalité de ce que je considérais avec tant d'attention, et qui m'a demandé tant d'efforts. Mais d'ailleurs, qui peut sérieusement douter de la réalité de ces objets mathématiques par excellence que sont les nombres entiers? Comment une chose, aussi concrètement et universellement utilisée pourrait-elle ne pas exister *matériellement*? La seule question qui se pose est : où se trouvent-ils? vous me direz : qu'on me montre l'entier 3! Je vais donc essayer maintenant de débusquer cet animal, là où il se cache.

Prenons tout d'abord un autre domaine des mathématiques, pour lequel, je crois, le travail

sera plus facile : les espaces vectoriels de dimension finie. A quelle réalité correspondent-ils? On a envie de dire qu'il s'agit de l'espace qui nous entoure, qui serait un espace vectoriel de dimension 3. Mais cette réponse ne tient pas longtemps : en relativité restreinte, c'est plutôt un espace de dimension 4, en relativité généralisée, c'est une variété de dimension 4, etc. Alors, où se trouvent vraiment les espaces vectoriels, les variétés, ... ? On répondra que ce ne sont que des représentations de l'espace qui nous entoure, et non pas l'espace lui-même. Ils n'existent donc réellement que dans notre tête. Mais sous quelle forme s'y trouvent-ils? Voilà la question essentielle ; la réponse nous fournira le concept fondamental pour toute la suite. Ce n'est pas pour rien qu'il est nommé deux fois dans le titre : c'est le concept de programme. Voyons cela un peu en détail, sur l'exemple des espaces vectoriels.

Nous avons admis que la notion d'espace vectoriel est un outil pour représenter l'espace dans lequel nous nous trouvons. Or, nous avons un outil évident pour nous repérer dans les trois dimensions de l'espace, ce sont nos deux yeux. En somme, la notion d'espace vectoriel doit être quelque chose, qui se trouve dans notre cerveau, et qui a un rapport avec nos yeux. De quoi s'agit-il donc?

Réponse : les yeux ne sont que les capteurs, qui envoient au cerveau l'information indispensable au repérage dans l'espace. Mais cette information, composée des millions de points colorés (qu'on appelle *pixels* en informatique) que les cellules de nos rétines envoient au cerveau, est encore sans structure, informe. Pour "voir" la profondeur, la hauteur et la largeur, le cerveau doit traiter cette masse de données à l'état brut au moyen d'un outil adéquat. Or, un outil de traitement de l'information, c'est exactement ce que l'on appelle un programme, ou un logiciel. Nous avons tous, dans le cerveau, ce programme qui traite l'information issue des yeux, et qui nous permet de "voir". C'est un logiciel remarquablement bien fait, les spécialistes d'intelligence artificielle essaient de l'imiter et n'arrivent qu'à de grossières approximations. Une expérience très simple montre d'ailleurs ce qui se passe quand ce programme ne fonctionne pas : vous connaissez sûrement ces images, formées de carrés colorés (de très gros pixels). Si on les regarde de près, elles n'ont aucun sens, on ne "voit" rien, car l'information reste non traitée, telle que les yeux la fournissent. Mais si on les éloigne un peu, un visage apparaît : le programme de la vision a fonctionné, il a traité l'information, et on voit une image.

Revenons à la notion d'espace vectoriel : sa nature devrait être claire maintenant. Il s'agit, évidemment, d'une (petite) partie de ce programme, ce qu'en informatique on appelle un *module de programmation*. Il en est de même des notions d'espace euclidien, de variété, etc.

Cette analyse soulève quelques objections immédiates. Tout en y répondant, j'en profiterai pour introduire quelques notions d'informatique, indispensables pour la suite.

D'abord une objection naïve : un chien ou un chat a, très certainement, un programme pour la vision qui est très voisin du nôtre. Pourtant, la notion d'espace vectoriel lui est probablement inconnue!

La réponse est simple : de toute façon, la notion d'espace vectoriel est inconnue à l'immense majorité du genre humain. C'est une chose de posséder un programme qui fonctionne, c'en est une autre de pouvoir l'écrire. Par exemple, lorsque vous vous servez d'une calculette ou d'un traitement de texte, vous disposez d'un programme en état de marche. Mais vous n'avez pas ce qu'on appelle "le source" du programme, c'est-à-dire son

texte écrit et commenté, qui n'est connu que de celui qui a programmé la machine que vous utilisez. C'est la différence, fondamentale en informatique, entre ce qu'on appelle "le source" (abrégé de "programme source") et le "code" qui est totalement incompréhensible, mais qui fait marcher la machine.
En résumé, tout le monde se sert de ses yeux, et donc du programme de la vision. Mais seuls ceux qui ont quelques notions de mathématiques connaissent un peu du "source" de ce programme, sous la forme, par exemple de la théorie des espaces euclidiens.

Une petite digression pour expliquer un peu plus précisément les termes informatiques de "source" et de "code" que je viens d'utiliser : le programme "source" est un texte tout à fait compréhensible, écrit dans un langage de programmation. Mais ce n'est pas lui qui est présent dans la machine : on le transforme d'abord en "code", qui est une suite de "bits", c'est-à-dire de 0 et de 1, et, de ce fait, absolument hermétique ; cette opération, qu'on appelle "compilation", est très facile à réaliser, à l'aide d'un programme qu'on appelle naturellement un "compilateur" (question en passant : comment a été compilé le compilateur?). La compilation s'apparente donc à un codage, c'est-à-dire à la transformation d'un message en clair en un message crypté. Bien entendu, son but n'était pas de crypter le texte du programme, mais simplement de le rendre utilisable par l'ordinateur ; mais le résultat est là. Du coup, l'opération inverse, de passage du "code" au "source", qu'on appelle "décodage" ou "décompilation", ressemble à celle qui consiste à casser un message secret. Elle est donc extrêmement difficile, et à vrai dire, carrément impossible la plupart du temps.
Passons à une deuxième objection : il existe, en mathématiques, des espaces vectoriels de grande dimension, 23 par exemple, ou même infinie. Ces espaces n'ont assurément rien à voir avec l'espace réel dans lequel nous évoluons. Dans ces conditions, pourquoi la notion générale d'espace vectoriel ferait-elle partie du programme de la vision?
La réponse est dans la notion informatique de "module de programmation", un terme que j'ai déjà utilisé plus haut. Voici ce dont il s'agit : quand on veut écrire un programme d'une taille imposante (et dieu, qui l'a écrit, sait que ça doit être le cas pour le programme de la vision), on le décompose en parties indépendantes, plus petites, qu'on appelle "modules". Chacun de ces modules est lui-même décomposé, et ainsi de suite. On arrive, de cette façon, à des programmes de taille raisonnable, que l'on peut écrire, parce qu'on peut les comprendre. L'avantage, aussi, si chacun de ces modules est bien commenté, est qu'ils peuvent resservir comme "briques" pour la construction d'autres programmes.
Or, chacun de ces modules a une portée beaucoup plus générale que ce qui est strictement nécessaire. Pour donner un exemple, supposons que vous ayez à résoudre numériquement une équation particulière, disons $x^3 - 7x + 1 = 0$. La façon la plus simple de procéder est de programmer une méthode générale, la dichotomie par exemple, valable pour toutes les fonctions continues. Vous créerez donc un tel module, que vous appliquerez seulement dans ce cas particulier. Mais, puisqu'il peut resservir pour une autre équation, vous le conserverez soigneusement dans la mémoire de l'ordinateur, dans ce qu'on appelle en informatique une "bibliothèque de programmes".
La théorie des espaces vectoriels est exactement l'un de ces modules de programmation, il fait partie d'une bibliothèque à laquelle le programme de la vision fait appel. Mais, bien entendu, il est évidemment réutilisé très souvent ailleurs.
Pour aller un peu plus loin, j'aurai besoin maintenant de donner quelques explications

simples sur la façon dont sont installés les programmes dans un ordinateur.
Très grossièrement, l'organisation des programmes dans une machine consiste en : une couche inférieure, appelée "système d'exploitation", ou "système" tout court, et une couche supérieure qui est formée des "programmes d'application"; on parle donc de "programmes système" par opposition à "programmes d'application".
Le système d'exploitation est un très gros programme (l'un des plus connus, le système UNIX, comporte actuellement plusieurs centaines de millions de caractères, et sa documentation occupe une vingtaine de gros volumes). Il gère les fonctions de base de la machine : le démarrage, les mémoires de masse (unités de disques durs, de bandes, de disquettes, ...), les communications entre la machine et ses utilisateurs (claviers et écrans), ses périphériques (comme les imprimantes, et les autres dispositifs que peut commander l'ordinateur, par exemple des robots) ; et aussi les communications avec les autres ordinateurs, à travers ce qu'on appelle un réseau.
Le système d'exploitation est, en général, livré avec la machine, car l'ordinateur ne peut pas fonctionner sans lui. C'est un programme qui "tourne" en permanence, mais que l'utilisateur courant ne voit pas.
Ce qu'il voit, ce sont les programmes d'application, qui sont écrits au niveau au-dessus : ce sont ces programmes qui permettent à l'ordinateur de se rendre utile. Par exemple, ce sont des programmes de comptabilité ou de gestion d'entreprise, ou encore de calcul mathématique, ou de traitement de texte, etc.
Ces programmes d'application démarrent à partir du système d'exploitation et y reviennent. L'utilisateur courant utilisera des programmes d'application tout faits. L'utilisateur plus expérimenté pourra écrire ses propres programmes d'application. Mais, sauf exception, il n'a pas accès au système d'exploitation : en effet, s'il est incompétent ou malveillant, il peut mettre l'ordinateur en panne, ou encore acquérir des informations qui ne lui sont pas destinées, et en faire un mauvais usage!
Les seuls utilisateurs ayant accès au système d'exploitation, qu'on appelle les "super-utilisateurs", sont normalement les ingénieurs système qui s'occupent de maintenir la machine en état de fonctionnement. Il y a une clé, ou un mot de passe, mis en place par celui qui a installé le système d'exploitation, clé dont la connaissance permet de "passer en super-utilisateur".
Or, comme chacun sait, tout ce qui est interdit provoque l'envie d'enfreindre l'interdiction. Ainsi sont apparus les "hackers" qui tentent de prendre le contrôle du système d'exploitation d'ordinateurs auxquels ils ne devraient pas avoir accès, dans le but de détruire ou d'acquérir de l'information, ou, tout simplement, pour s'amuser.
Le mathématicien se trouve dans une situation tout à fait comparable à celle du "hacker" : il est en face d'une machine (le cerveau, le sien mais aussi celui des autres) qui lui a été livrée en état de marche, donc avec son système d'exploitation, mais sans documentation et sans clé! Il se trouve relégué au simple niveau d'utilisateur courant, et encore ne peut-il utiliser que des programmes tout faits, il n'a même pas le droit de programmer la machine. Situation tout à fait intolérable et humiliante. Comme tout bon "hacker", son but ultime est de se hisser au rang de "super-utilisateur". Comment faire? On devine que ça ne va pas être de la tarte, et, disons-le tout de suite, pour vous épargner le suspense, ce but ultime n'est pas encore atteint. Mais, à mon avis, on n'en est peut-être pas si loin.
Quand on se trouve devant une machine inconnue, sans aucune documentation, le seul

moyen d'apprendre à s'en servir et à la programmer (sans parler d'arriver à maîtriser son système d'exploitation), c'est de s'armer de patience et la faire fonctionner, en la surveillant attentivement pour décoder, petit à petit, les programmes qui s'y trouvent (programmes d'application, ou programmes système). C'est plus difficile à déchiffrer que les hiéroglyphes, car, non seulement on n'a pas de pierre de Rosette, mais on ne sait même pas dans quels caractères c'est écrit ...

Le mathématicien s'est donc armé d'une longue patience – on peut bien le dire, puisque qu'il travaille depuis plus de trois mille ans. Il a commencé à décoder des programmes d'application, on en a vu un exemple tout à l'heure avec la théorie des espaces vectoriels. La moisson est maintenant fantastiquement riche et variée : cela a commencé avec la géométrie, puis l'algèbre, puis l'analyse réelle, les fonctions de variable complexe, la théorie des nombres, et enfin le développement explosif des mathématiques au XXème siècle.

Poursuivons notre comparaison entre le mathématicien et le "hacker", pour la préciser un peu. Habituellement, le "hacker" travaille seul, et s'attaque à une seule machine. Son outil, c'est tout simplement un clavier et une console, branchés sur le réseau, ou directement sur l'ordinateur qui l'intéresse. Son arme, c'est sa connaissance plus ou moins approfondie du système d'exploitation de la machine qui l'intéresse.

Mais on peut imaginer un gang de "hackers", s'attaquant simultanément, chacun de leur côté, à des machines identiques et se communiquant, au fur et à mesure, le résultat de leurs essais, c'est-à-dire ce qu'ils ont appris sur la façon dont est programmée la machine. Leur efficacité deviendrait alors vraiment redoutable. S'ils sont assez nombreux, ils pourraient publier des articles sous le manteau et tenir des colloques clandestins. Pour verser carrément dans la fiction, imaginons que le travail de ces pirates s'étale sur plusieurs générations, les parents léguant à leurs enfants leurs résultats et le soin de poursuivre l'attaque jusqu'à la victoire finale. Ce scénario invraisemblable est pourtant une image, à peine caricaturée, de ce que nous faisons, nous les mathématiciens ; pas étonnant que l'on nous prenne souvent pour de doux dingues.

Mais on peut maintenant se demander où sont le clavier et la console qui permettent l'accès à la "machine" qui nous intéresse ici, à savoir le cerveau, et de faire "tourner" les programmes que l'on a réussi à décoder. La réponse à cette question nous expliquera aussi pourquoi, à la différence du "hacker", le mathématicien ne peut travailler seul. Mais il faut d'abord dire un mot sur ce qu'on appelle un "réseau d'ordinateurs".

Il s'agit d'un ensemble d'ordinateurs, reliés entre eux par cable, par téléphone, ou par radio, et qui peuvent échanger ainsi des informations, c'est-à-dire des textes ou des programmes. A partir de l'un quelconque d'entre eux, on peut donner des ordres à n'importe quel autre. C'est extrêmement utile, car cela permet à chaque utilisateur d'avoir accès à la puissance de calcul, aux mémoires de masse et aux périphériques de chacune des machines du réseau. Le réseau peut être local, par exemple à l'échelle d'une entreprise, d'un laboratoire ou d'une université. Mais il y a aussi un réseau à l'échelle mondiale, dont l'utilisation la plus courante est le courrier électronique, mais qui permet tous les usages dont je viens de parler.

Bien entendu, le système d'exploitation de chaque machine comporte des programmes spécialisés très complexes, qui servent à la communication à travers le réseau ; on les appelle des "protocoles de communication". Il faut comprendre que le dialogue entre

deux ordinateurs est très protocolaire, et qu'à la moindre formule de politesse oubliée, l'ordinateur offensé coupe immédiatement la communication. Ces formules de politesse sont, en effet, indispensables pour s'assurer que la ligne fonctionne bien, que chacun est prêt à communiquer avec l'autre, etc ; bref, que l'information pourra être transmise absolument sans aucune erreur, ce qui est essentiel lorsque l'on transmet des programmes (alors que quelques erreurs sont sans importance quand on transmet un texte).
Le "réseau des mathématiciens", si l'on peut dire, s'est constitué au cours des siècles. Les moyens physiques de communication étaient les mêmes que pour les autres activités humaines, c'est-à-dire la parole et le texte écrit. Mais les mathématiciens ont créé, ou plus exactement découvert, petit à petit, leur protocole de communication : ainsi est apparu le langage mathématique, avec son étrange litanie d'axiomes, de lemmes, de propositions, de théorèmes et de démonstrations. La communication mathématique ne peut se faire qu'à travers ce protocole, dont le rôle est bien d'assurer la transmission de l'information sans aucune erreur. Il permet à un mathématicien d'essayer, sur un collègue bénévole et consentant, de faire "tourner" un programme (c'est-à-dire un théorème avec sa preuve), qu'il a découvert, ou plus exactement décodé, dans son propre cerveau. Pour cela il faut (et il suffit!) que le collègue en question veuille bien faire l'effort de comprendre la preuve ; les signes de sa compréhension sont suffisamment manifestes pour qu'il ne puisse y avoir de doute (grognements approbatifs, remarques pertinentes, ou même amélioration de la preuve, ou du résultat, etc). C'est ainsi que l'on s'assure que le programme que l'on a reconstitué est bien valable, et qu'il peut aller s'ajouter à la somme des mathématiques déjà découvertes.
Résumons-nous : un protocole de communication est un programme dont le but est de permettre l'échange d'information entre deux ordinateurs. Les mathématiciens ont découvert, dans le cerveau, un programme de ce type, ou qui pouvait servir à cela, et l'ont fait fonctionner : c'est le langage mathématique, qui a fini par aboutir à ce qu'on appelle la méthode axiomatique.

Mais alors, cela devait arriver, quelques-uns parmi nos chercheurs en mathématiques se sont dit que ce programme était tout aussi intéressant, et peut-être même plus, que d'autres, comme objet d'étude. C'est la naissance d'une nouvelle branche des mathématiques : la logique mathématique, et, pour être plus précis, la théorie de la démonstration. Son but est donc d'essayer de décoder ce protocole de communication.
Nous arrivons maintenant à un point très intéressant, mais que je ne développerai pas ici. On doit remarquer, en effet, qu'un protocole de communication est ce qu'on appelle un programme de bas niveau dans le système d'exploitation, c'est-à-dire qu'il fait partie du mode de fonctionnement le plus primitif de l'ordinateur. Cela ne veut pas dire qu'il soit simple, mais plutôt qu'il est situé dans les couches primaires du système, et, qu'à ce titre, il y occupe une place stratégique. On est donc amené à penser que la théorie de la démonstration s'occupe de décoder des programmes très primitifs dans le cerveau, beaucoup plus que ceux de la vision, de l'audition, de l'équilibre, etc. Ces programmes sont donc certainement apparus très tôt dans l'évolution, et doivent être communs à un grand nombre d'espèces. Mais, bien entendu, il n'y a que l'homme qui les ait utilisés comme protocoles de communication.
Un autre point remarquable est le suivant : c'est en déchiffrant ces programmes, c'est-à-dire en faisant de la théorie de la démonstration, que les mathématiciens ont commencé

à prendre vraiment conscience de la nature de leur activité. En effet, c'est alors qu'a été découvert ce que les logiciens appellent "l'isomorphisme de Curry-Howard", qui dit essentiellement qu'une preuve mathématique n'est pas autre chose qu'un programme. Il donne aussi le moyen d'exécuter ce programme, qui consiste à faire, sur la preuve, une opération appelée "élimination des coupures". Il s'agit là d'une importante découverte de logique, et il se se constitue actuellement un dictionnaire, où chaque notion de théorie de la démonstration se trouve traduite en termes de programmation. C'est une équivalence tout à fait étonnante, dont voici quelques exemples:

Théorie de la démonstration		Programmation
Règle logique	$\Leftrightarrow$	Instruction
Démonstration	$\Leftrightarrow$	Programme
Elimination des coupures d'une démonstration	$\Leftrightarrow$	Exécution d'un programme
Preuve par récurrence	$\Leftrightarrow$	Boucle "for"
Théorème	$\Leftrightarrow$	Type, spécification
Axiome	$\Leftrightarrow$	Déclaration de variable
Règle de l'absurdité (de "faux", déduire n'importe quoi)	$\Leftrightarrow$	Instruction "exit"

Bien sûr, tout ceci nécessiterait des explications détaillées, qu'il n'est pas possible de donner ici.
Mais, à propos, il est temps de se rappeler la promesse que je vous ai faite, de vous montrer l'entier 3. Il doit donc s'agir, comme pour toutes les notions mathématiques, d'un module de programmation, d'un élément, particulièrement simple, de notre grande bibliothèque de programmes. Lequel? Si l'on sait qu'en mathématiques, la notion d'entier s'identifie au raisonnement par récurrence, l'isomorphisme de Curry-Howard nous donne la réponse : c'est la boucle "for i = 1 to 3", c'est-à-dire un programme qui consiste à exécuter trois fois le programme qui le suit. Une telle instruction élémentaire fait inévitablement partie d'un langage de programmation, quel qu'il soit.

Revenons à notre analyse de l'activité mathématique, pour résumer ce à quoi nous sommes parvenus : les concepts, les théories et les théorèmes mathématiques ne sont pas autre chose que des programmes, plus précisément des "modules de programmation", qui existent dans notre cerveau, mais, bien entendu, aussi dans celui des animaux qui sont proches de nous (les mammifères, par exemple).
Cette approche a aussi l'avantage de rendre compte du fait que les mathématiques soient utiles : en effet, elles permettent de coordonner l'action d'un grand nombre de personnes de façon très précise, puisque tout le monde les comprend de la même façon (ou ne les comprend pas du tout). C'est le propre d'un programmme de donner toujours le même résultat, quand, par chance, on arrive à le faire fonctionner.

Tout cela est bel et bon, me direz-vous, mais voici une objection imparable à cette analyse : c'est l'adéquation, quasiment miraculeuse, entre les mathématiques et les "lois de la nature", par exemple les lois de la physique. Comment se fait-il, en effet, que la loi

d'attraction de Newton, ou la mécanique quantique, nous permette de prévoir des évènements astronomiques, ou le résultat d'expériences à l'échelle atomique, avec une précision extraordinaire?

En fait, l'explication est simple, justement si l'on admet que les mathématiques ne sont que le décodage de programmes "utilitaires" que dieu, autrement dit un milliard d'années d'évolution, a écrits dans le cerveau des êtres vivants supérieurs. Il n'y a alors vraiment rien d'étonnant à ce que ces programmes soient en profond accord avec l'environnement de ces êtres vivants ; sinon, ils rempliraient bien mal leur rôle, qui est d'adapter le mieux possible l'organisme qu'ils régissent à cet environnement. Par suite, rien d'étonnant non plus à ce que, en lisant ces programmes dans notre propre cerveau, nous croyions découvrir des "lois de la nature".

En d'autres termes, et pour mettre les points sur les i, les lois de la physique, comme l'attraction newtonnienne, la relativité générale ou la mécanique quantique, sont bien écrites, sous forme de programmes, dans le cerveau d'un chien ou d'une vache (soit dit en passant, et heureusement pour elle, il y a des choses autrement plus complexes dans le cerveau de la vache). En faisant de la physique mathématique, des hommes ont réussi à décoder ces programmes, et donc à les faire fonctionner selon un mode bien différent de celui pour lequel ils ont été écrits (et cela, malheureusement pour elle, la vache en est incapable).

Essayons de donner un autre éclairage, au moyen d'une caricature assez grossière. Imaginons que vous ayez à programmer un robot, de façon à ce qu'il puisse se déplacer dans une pièce, sans se heurter aux meubles ni aux murs. Vous commencez par écrire votre programme en fonction d'une pièce donnée, par exemple celle où vous vous trouvez maintenant. Mais, s'il passe dans la salle à côté, notre robot aura perdu son adaptation et se cassera immanquablement la figure. Qu'à cela ne tienne, on en prend un autre, et vous modifiez votre programme en lui ajoutant une partie destinée à tenir compte de la nouvelle pièce. Ce faisant, vous vous rendez compte qu'il est déraisonnable d'essayer d'écrire un programme pour chaque pièce possible, jamais vous n'aurez le temps, jamais ils ne tiendront dans la mémoire du robot. En plus, vous allez casser trop de matériel, avec vos essais!

Vous reprenez alors votre travail de programmeur sur de nouvelles bases, en vous demandant ce qu'on peut dire de général sur toutes les pièces où pourra se trouver votre robot ; et vous inventez des lois : les pièces sont des parallélépipèdes rectangles, les meubles aussi, le plafond est à au moins deux mètres, etc. Ces lois, qui forment une espèce de géométrie, vont, bien sûr, se retrouver inscrites, sous forme de modules, dans le programme final.

Si maintenant un autre programmeur, ou le robot devenu subitement intelligent, se met à décoder votre programme, il va évidemment y trouver ces lois, qu'il baptisera "lois de la nature", et, naïvement, il s'émerveillera de leur extraordinaire similitude avec l'environnement. S'il avait décodé votre premier programme mal fait, il n'aurait rien trouvé du tout. Mais il n'a jamais eu l'occasion de le faire, car votre premier robot avait été bien trop vite mis à la casse.

Je vais dire la même chose d'une autre façon encore ; comme vous le voyez, j'insiste car ce point est important. La physique mathématique n'explore pas notre environnement actuel, mais la trace qui en a été déposée dans notre cerveau au cours des âges par l'évolution. Comme Champollion, qui déchiffrait des hiéroglyphes qui parlent d'un passé

très lointain, le physicien théoricien déchiffre des programmes écrits dans le cerveau il y a des dizaines de millions d'années. Ces programmes expriment certainement des lois très durables de l'environnement, car elles doivent être restées les mêmes tout ce temps-là : sinon, l'organisme dont ces programmes règlent la vie aurait perdu son adaptation, et nous ne serions pas là pour en parler. C'est bien pourquoi la physique mathématique exprime des lois "universelles" de la nature. Mais, de même que l'historien, ou l'archéologue, pour écrire l'histoire, ne peut créer aucun document, et doit se contenter d'exploiter ceux qui existent déjà, de même le physicien ne découvrira jamais, comme loi de la nature, que ce que l'évolution a bien voulu en écrire dans notre cerveau. A vrai dire, qu'il se rassure, il lui reste quand même du pain sur la planche.

Je terminerai sur une remarque à propos de la méthode utilisée ici pour analyser la nature de l'activité mathématique. Elle paraîtra à beaucoup simpliste et mécaniste, puisqu'elle consiste à comparer le cerveau à un ordinateur. Or, tout le monde vous le dira, le cerveau est très loin de tout ordinateur existant, et il ne fonctionne absolument pas de la même manière. Je pense qu'en disant cela, on fait une lourde erreur, parce qu'on oublie un détail important : un ordinateur, de toutes façons, ne fonctionne pas du tout. Il n'est pas autre chose qu'une page blanche, il n'existe que par les programmes qu'on y écrit. Or ce sont des hommes qui l'ont programmé, et ils ont fait, consciemment ou non, des efforts gigantesques pour que son fonctionnement ressemble le plus possible à celui de leur propre cerveau. Pourquoi cela?
1) parce que c'est la seule façon de rendre la machine utile à quelque chose, et 2) parce qu'ils ne savent pas quoi faire d'autre. Il paraît qu'on en est actuellement à la quatrième ou la cinquième génération d'ordinateurs, et on peut alors penser que la simulation commence à ne pas être si mauvaise. En fin de compte, ce n'est pas notre cerveau qui ressemble à un ordinateur, mais c'est l'ordinateur qui essaie à toutes forces, par programmeur interposé, de ressembler à notre cerveau.
Personne ne dira que, puisqu'un livre est très différent d'un homme, on ne peut rien apprendre sur l'homme en lisant un livre. Je prétends qu'il y a déjà beaucoup, et qu'il y aura de plus en plus, à apprendre sur le fonctionnement du cerveau humain en lisant dans un système d'exploitation ou même dans le microcode d'un processeur.

****