Le courrier du CNRS 80

Auteur(s) : CNRS

Les folios

En passant la souris sur une vignette, le titre de l'image apparaît.

116 Fichier(s)

Les relations du document

Ce document n'a pas de relation indiquée avec un autre document du projet.

Citer cette page

CNRS, Le courrier du CNRS 80, 1993-02

Valérie Burgos, Comité pour l'histoire du CNRS & Projet EMAN (UMR Thalim, CNRS-Sorbonne Nouvelle-ENS)

Consulté le 12/08/2025 sur la plate-forme EMAN : https://eman-archives.org/ComiteHistoireCNRS/items/show/170

Présentation

Date(s)1993-02

Editeur de la ficheValérie Burgos, Comité pour l'histoire du CNRS & Projet EMAN (UMR Thalim, CNRS-Sorbonne Nouvelle-ENS)

Information générales

LangueFrançais
CollationA4

Informations éditoriales

Directeur de la publicationKourilsky, françois
N° ISSN0153-985x

Description & Analyse

Personnes citéesGagnepain, Jean-Jacques
Nombre de pages116
Notice créée par Valérie Burgos Notice créée le 05/10/2023 Dernière modification le 24/12/2024

LE COURRIER DU CNRS

DOSSIERS SCIENTIFIQUES

LA RECHERCHE EN INFORMATIQUE

Les nouveaux ordinateurs
Programmer aujourd'hui et demain
Informatique théorique et mathématiques
L'intelligence artificielle
La robotique avancée
La machine communicante

N° 80 - 50F - FEVRIER 1993

COUVERTURE

Circuit intégré vu en microscopie électronique à balayage (© P. Arnold - FOVEA).

En 4e de couverture

Illustration G. Ouanounou - INRIA.

LE COURRIER DU CNRS
DOSSIERS SCIENTIFIQUES

Directeur de la publication :
François Kourilsky

RÉALISATION
CNRS-Atelier de l'Ecrit
Groupement des Unités de la Communication du CNRS
1, place Aristide-Briand
92195 Meudon Cedex

Réduction en chef : Sylvie Langlois
Rédaction : Pierrette Massonnet, Christine Girard
Secrétariat : Muriel Hourlier

Ce numéro du *Courrier du CNRS* a été préparé sous la direction du département scientifique Sciences pour l'ingénieur du CNRS.

Prix : 50 francs.
Vente au numéro : CNRS - Éditions, 20-22, rue Saint-Amand, 75015 Paris - tél : (1) 45 33 16 00.

Le Courrier du CNRS remercie les auteurs et les organismes qui ont participé à ce dossier.

Fabrication
Direction artistique : Gris Bleu - tél : (1) 43 45 98 00
Photogravure : SRO - tél : (1) 43 45 98 00
Impression : Roto-France Impression
tél : (1) 60 06 60 00
Commission paritaire : AD 303.
ISSN : 0153-985X. ISBN : 2-222-04765-X.

UN NOUVEAU PROJET POUR LE COURRIER DU CNRS

Ce numéro 80 du *Courrier du CNRS* consacré à la recherche en informatique est le dernier à paraître sous cette forme. Une réflexion est actuellement menée au sein de la Mission de la communication et de l'information scientifique et technique, en concertation avec les directions des départements scientifiques, afin d'arrêter un nouveau projet éditorial.
Prochain rendez-vous au cours du second semestre 1993.

EDITORIAL

Jean-Jacques Gagnepain,
Directeur scientifique
du département des Sciences
pour l'ingénieur du CNRS

Le phénomène informatique est sans précédent dans l'histoire : en moins d'un demi-siècle, il aura bouleversé le paysage socio-économique et marqué des concepts de base du savoir.

L'informatique représente 10 % des investissements hors bâtiments des sociétés françaises ; parmi les cadres techniciens, on dénombrait 50 000 informaticiens au recensement de 1982, 150 000 à celui de 1991. Moins visibles mais aussi vitaux, les réseaux informatiques succèdent aux châteaux de l'industrie. Les sociétés découvrent le traitement de l'information après avoir appris à transformer la matière et l'énergie, et on peut comparer la révolution informatique à celle de la machine à vapeur.

Il n'est donc pas besoin de mettre l'accent sur les applications de l'informatique, car elles sont partout. Mais les rapports entre l'informatique, ses concepteurs et ses utilisateurs, changent. L'informatique est de plus en plus conviviale, on parle de machine partenaire, de machine intelligente. L'ergonomie, la multifonctionnalité, l'intégration à la société, l'adaptabilité, la sûreté de fonctionnement – autant d'attributs qui favorisent la banalisation dans l'esprit de l'usager – sont le fruit d'efforts soutenus des concepteurs, de progrès de la discipline reposant sur des mathématiques et de la physique de base, aussi bien que sur des technologies et des collaborations interdisciplinaires.

L'informatique porte aussi une formidable ambivalence : comme industrie et technologie, elle est omniprésente ; comme science de l'information, elle change nos rapports au monde. Depuis l'Antiquité, en passant par la Renaissance et les Lumières, les automates fascinent l'homme. La magie du mime demeure, mais les interrogations qu'elle sous-tend ont glissé du mythe à la raison. De l'ordinateur au cerveau, du robot à l'homme, la métaphore est incontournable, même si elle est discutable et tributaire d'une imagerie technologique où l'électronique (ou la dynamique moléculaire) remplace l'horlogerie. Et peut-être que les profonds résultats de limitation en calculabilité et logique, fondés sur le discret et une modélisation de la physique à notre échelle, seront dépassés par des progrès en microphysique qui feront à nouveau exploser le savoir et le savoir-faire. En attendant, les changements d'échelles dans les applications et simulations actuelles permettent d'entrevoir un contrôle de plus en plus pénétrant de notre environnement et de l'impact de nos actions.

Interdisciplinaire par excellence dans un département scientifique interdisciplinaire par vocation, l'informatique figure en bonne place dans les priorités du département des Sciences Pour l'Ingénieur et dans la stratégie du CNRS - sans parler des activités de service. Le CNRS développe un partenariat avec les autres grands organismes, en particulier l'INRIA, et il dispose de 200 chercheurs permanents en informatique, autant d'ITA, regroupés aux côtés d'un millier d'universitaires dans une quarantaine d'unités ou de groupements de recherches.

Le CNRS se devait d'être en première ligne dans cette aventure économique, sociale et culturelle, dont ce dossier présente les principales facettes. Et l'effort sera intensifié, car on augure que, portée par les progrès technologiques et les demandes nouvelles, la croissance va se poursuivre pendant des lustres au rythme de ces dernières décennies.

SOMMAIRE

L'INFORMATIQUE AUJOURD'HUI

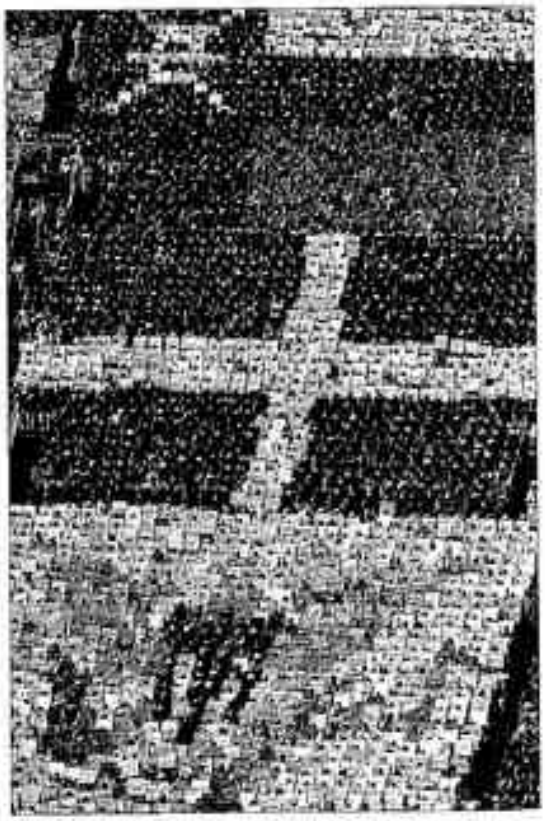
Jean-Sylvain Liénard
Jean-Pierre Verjus

Ce numéro du *Courrier du CNRS* dresse un panorama de l'informatique d'aujourd'hui. L'informatique s'est affirmée au cours des ans comme une discipline à part entière, avec ses fondements, ses méthodes et ses outils spécifiques. Elle s'est modelée sous une triple poussée : celle de la technologie, qui au niveau matériel fournit une puissance de calcul toujours croissante pour un prix qui diminue tous les jours, et au niveau logiciel des systèmes de plus en plus sophistiqués et mondialement interconnectés ; celle de l'utilisateur, qui aimerait voir la machine comme un véritable partenaire intelligent, actif, capable de percevoir le monde et de s'exprimer ; celle des avancées conceptuelles enfin, qui permettent de mettre la technologie au service de l'utilisateur, en fonction de ce qu'on sait être calculable par un algorithme et avec les moyens d'expression maîtrisés du moment (un langage de programmation).

L'informatique couvre aujourd'hui un vaste domaine, allant de la conception de circuits à l'intelligence artificielle, des interfaces homme-machine à la logique formelle, en passant par l'algorithmique et la programmation. Elle est en interaction avec de nombreuses autres disciplines, qu'il s'agisse des mathématiques, de l'électronique, de l'optique ou des sciences cognitives, au point qu'il est parfois difficile d'en tracer précisément les contours. Elle reste cependant bien ancrée dans les Sciences de l'Ingénieur par son aspect pragmatique, par l'universalité de ses méthodes et de ses outils, par ses rapports avec le réel, par sa capacité de validation des modèles.

Jusqu'à présent, l'informatique a été largement déterminée par les avancées technologiques. Les chercheurs essaient de relever le défi de construire des machines composées de plusieurs milliers de processeurs, capables d'exécuter chaque seconde des milliers de milliards d'opérations ; de telles machines sont nécessaires pour mener à bien certaines applications comme la prédiction météorologique à quatre ou cinq jours avec une très bonne précision. Les difficultés majeures se situent au plan architectural, c'est-à-dire au niveau du contrôle des processeurs et des échanges mutuels d'information. Des problèmes semblables se posent lorsqu'on considère l'ensemble des machines connectées sur les réseaux locaux, nationaux et internationaux, que l'on tend de plus en plus à considérer comme un système gigantesque offrant des capacités de calcul et de mémorisation quasi illimitées. L'analogie avec le cerveau, fait de multiples organes de calcul, les neurones, est tentante mais sans doute prématurée. Il apparaît ainsi difficile de promouvoir des machines fondamentalement différentes de celles définies par Von Neumann, parce que c'est techniquement difficile, parce que cette approche ne montre pas de limite évidente et parce qu'enfin les difficultés majeures sont actuellement plus aux niveaux de l'algorithmique, de la programmation et des processus de communication homme-machine, qu'au niveau de l'architecture des systèmes.

La puissance des machines est désormais telle que les problèmes de logiciel ont complètement changé de nature. Programmer n'est plus un exercice couplé à la découverte d'un algorithme, lorsqu'en faisant telle astuce de programmation, on gagnait en volume de mémoire ou en temps d'exécution. Il s'agit avant tout de maîtriser la complexité, de construire des logiciels sûrs, modulaires, extensibles, dont le comportement soit prévisible dans tous les cas et qui puissent



Une vue des jeux olympiques de Los Angeles en 1984 : l'élaboration d'une configuration collective à partir de comportements individuels est en rapport avec certaines notions informatiques de base, comme le parallélisme, la synchronisation, le calcul distribué, les réseaux d'automates (© G. Rancinan - Sygma).

être aisément communicables et maintenus. Les grands logiciels de contrôle de centraux téléphoniques, d'avions, de systèmes d'armes atteignent couramment des millions d'instructions ; pour les réaliser et les valider, on ne peut procéder par test de tous les cas de figure possibles : il faut mettre en œuvre des méthodes rigoureuses de conception et de réalisation, dites méthodes d'ingénierie du logiciel. De ce fait, le logiciel est désormais une technologie, avec ses normes, ses pesanteurs et l'obligation, lorsqu'on veut mettre sur le marché une nouvelle génération, de la rendre compatible avec la génération précédente. Bien entendu, la recherche sur de nouveaux moyens d'expression des algorithmes (les recherches sur de nouveaux langages de programmation, par exemple) n'en garde pas moins tout son intérêt, mais ne doit plus être confondue avec la technique de production de logiciel aux normes industrielles.

La problématique de l'informaticien est double : d'une part, il doit pouvoir répondre à la question "est-ce calculable ?" et savoir évaluer la complexité de ce calcul, c'est-à-dire son coût ; et d'autre part, il doit donner à l'utilisateur la possibilité de s'exprimer dans le langage de son choix, selon sa culture et la nature du problème traité. Il doit aussi se préoccuper des langages et de leur sémantique, c'est-à-dire de leur pouvoir "d'expression".

Exprimer la requête suivante à un ordinateur : "quels rectangles peut-on former avec neuf carrés dont les côtés sont des nombres entiers tous différents ?" est désormais possible. Entre ce langage et celui qui est reconnu par la machine la plus simple qu'on ait formalisée – les automates d'états finis – la distance est considérable. C'est un défi majeur de l'informatique que d'essayer de traduire l'expression d'un problème pas toujours très facilement formalisé, à une machine qui ne connaît qu'un langage rudimentaire et ne supporte aucun non-dit. Un autre défi est de fournir des outils généraux d'analyse d'algorithme, pour évaluer le temps d'exécution et l'encombrement du programme qui le met en œuvre. Mais nombre d'informaticiens sont amenés à étudier des algorithmes nouveaux, en particulier lorsqu'il s'agit de problèmes limites, très difficiles (en théorie des graphes, en cryptographie, en géométrie algorithmique), voire impossibles à résoudre avec des méthodes déterministes (en recherche opérationnelle) ou tels que leur étude ne peut pas être faite indépendamment d'une connaissance assez approfondie des machines, comme dans l'algorithmique parallèle.

L'objet principal de l'intelligence artificielle (IA) est le traitement de la connaissance exprimée sous forme symbolique : ce peut être un objet du monde physique ou un objet immatériel propre à la cognition humaine, ce peut être aussi la relation formelle entre deux objets.

Au centre de l'IA on trouve les notions de représentation des connaissances, de raisonnement et d'apprentissage. La logique mathématique y joue un rôle important, mais s'avère insuffisante dès que l'on n'est plus dans un cadre strictement déductif. Ainsi apparaissent diverses extensions, appelées "logiques non classiques", qui traduisent en fait la difficulté de traiter des données du monde réel, floues, contradictoires, incertaines - dont pourtant l'esprit humain s'accommode fort bien ! D'où le lien de plus en plus fort qui s'établit entre l'IA, la linguistique, la psychologie et les autres sciences cognitives (voir le numéro 79 du *Courrier du CNRS*). La notion d'apprentissage automatique, reposant sur l'idée qu'il vaut peut-être mieux donner à une machine la possibilité d'apprendre elle- même, plutôt que de lui injecter de la connaissance, est également en forte expansion : en somme, mieux vaut une machine bien faite qu'une machine bien pleine !

La robotique contemporaine est, comme la communication homme-machine, en étroite symbiose avec l'IA. Dans les deux cas, il s'agit de passer des signaux en provenance du monde réel à leur représentation sous forme de symboles affectés d'une signification, de traiter ces représentations en fonction d'une certaine tâche à accomplir, de générer des signaux de commande vers l'extérieur. En robotique, les signaux à percevoir sont fournis par des capteurs spécialisés qui ne sont pas nécessairement analogues aux organes sensoriels humains ; la connaissance mémorisée, nécessaire à l'interprétation des signaux et à l'élaboration d'un certain comportement, est de nature physique. La robotique fait une large part à la mécanique, l'automatique, l'élec-

Le verger Beuri : image de synthèse mettant en œuvre un modèle botanique de la croissance des plantes, régi par des paramètres agronomiques comme le rythme des saisons ou les attaques de parasites (cliché CIRAD - GERDAT, Montpellier).

tronique, mais est principalement liée à l'informatique ; elle met en relief des notions comme le temps réel, la fusion de senseurs, la planification, la réactivité, qui en retour constituent des sujets de réflexion pour l'informatique tout entière.

Même autonome, un robot doit avoir une capacité de communication avec l'opérateur. On retrouve en communication homme-machine les mêmes fonctions de perception (analyse, reconnaissance), traitement des connaissances, génération (synthèse) ; mais ici, il s'agit de signaux émis ou perçus par des humains. Certains de ces signaux sont organisés selon un langage, écrit ou parlé, dont les structures sont parfaitement adaptées au fonctionnement de l'esprit humain. D'autres signaux, comme certaines formes d'images, sont porteurs de signification sans pour autant relever de structures langagières. L'opérateur humain est capable d'intégrer tous ces signaux dans un ensemble cohérent et signifiant : la machine doit être capable d'en faire autant, pour pouvoir soutenir un véritable dialogue avec lui, dans la perspective de l'accomplissement d'une certaine tâche. Ici il apparaît clairement que l'ordinateur est en passe de devenir plus qu'un outil : un véritable partenaire, intelligent, puissant, coopératif, à qui ne manquera même pas la parole ! Au terme de ce tour d'horizon, il nous faut rappeler que l'informatique, qui se développe simultanément et rapidement par l'amélioration de sa technologie matérielle et logicielle, par l'approfondissement de ses fondements théoriques, et par une meilleure interaction avec le monde réel, physique et humain, reste profondément unitaire dans son esprit même. Il n'y a pas de hiatus dans ce vaste panorama ; tous les aspects sont liés les uns aux autres, et les informaticiens continuent de faire preuve d'une créativité particulière, parce que l'ordinateur est une machine puissante, certes, mais au départ complètement ignorante. Il faut tout lui apprendre, il faut tout repenser pour elle, et cela oblige l'informaticien, quelle que soit sa spécialité, à porter un regard neuf sur nombre de questions que l'on considérait autrefois comme triviales ou allant de soi. Ce faisant, il est souvent amené à porter un regard neuf également sur lui-même et sur son activité cognitive. Ainsi l'informatique est à la fois objet d'investigation scientifique, outil universel, catalyseur des autres sciences et miroir de l'esprit humain.

Jean-Sylvain Liénard, directeur de recherche au CNRS, Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur.

Jean-Pierre Verjus, professeur à l'Institut national polytechnique de Grenoble, directeur de l'IMAG (Fédération de laboratoires CNRS à Grenoble).

UN PEU D'HISTOIRE

Au début des années 50, il n'y avait dans le monde, pour l'essentiel aux Etats-Unis, en Grande-Bretagne et en France, que quelques prototypes de machines à calculer, mettant en œuvre les principes architecturaux définis par Von Neumann. L'idée de base était de mettre dans une même mémoire non seulement les données à traiter, mais aussi les codes décrivant la suite d'instructions à exécuter, ce qui permettait de construire des séquences complexes d'opérations arithmétiques et logiques, et d'en maîtriser le déroulement en fonction de résultats intermédiaires. Les premières machines disponibles commercialement étaient chères et d'accès difficile. La programmation au début des années 60 était affaire de spécialistes connaissant à fond la structure des machines. On vit ensuite se développer des langages évolués comme COBOL et FORTRAN qui mettaient les machines à la portée des utilisateurs des premières grandes disciplines concernées (la gestion et le calcul scientifique), et les premiers fondements de l'informatique qui donnaient des bases théoriques à la calculabilité et à la programmation, préparant les langages et algorithmes de la génération suivante.

Les années 60 furent celles de la "grande informatique" : les machines étaient placées dans de grands centres de calcul, où les utilisateurs amenaient leur paquet de cartes perforées et venaient récupérer leur "listing" de résultats. Dans les années 70, la technologie des circuits imprimés et des mémoires a permis de développer des machines moins coûteuses, appelées mini-ordinateurs, qui se répandirent très vite dans les entreprises, les laboratoires et les administrations. C'est à cette époque que notre pays prit conscience qu'il fallait à l'informatique une formation spécifique et non simplement quelques cours supplémentaires introduits dans les formations scientifiques traditionnelles. L'informatique avait changé, le rôle de l'informaticien ne se limitait plus à programmer mais évoluait vers l'analyse des problèmes, la communication avec les utilisateurs et l'utilisation optimale des machines et systèmes qu'on commençait à interconnecter.

A la mini-informatique succéda la micro-informatique des années 80, qui fit entrer l'ordinateur dans tous les secteurs de la vie courante. Aujourd'hui, l'informatique est individuelle, distribuée, omniprésente. C'est une activité de masse, un fait social et culturel. Il s'ensuit d'immenses besoins de rationalisation au niveau des matériels, des méthodes, des langages par exemple, et d'ouverture vers le monde physique représenté par d'autres machines informatiques ou non, et vers l'homme avec lequel on cherche une communication de plus en plus naturelle et efficace.

Les nouveaux ordinateurs

Connection Machine CM-5, le dernier supercalculateur de Thinking Machines Corporation (cliché S. Grohel).

Le terme "architecture des ordinateurs" désigne traditionnellement l'ensemble des activités concernant la conception et la réalisation des éléments matériels des systèmes informatiques. Mais aujourd'hui ce terme a un sens plus large et il s'étend aux activités de conception et de réalisation des outils de base, aussi bien logiciels que matériels, qui servent de support aux applications dans les systèmes informatiques. On verra que l'extension du sens du mot "architecture" n'est pas le fait du hasard, mais la conséquence logique de l'interpénétration des fonctions logicielles et matérielles dans les nouveaux ordinateurs.

L'architecture des ordinateurs a quarante ans. En effet, le concept d'architecture est étroitement lié à la venue, à la fin de la deuxième guerre mondiale, du premier calculateur électronique programmable (ou ordinateur) qui a été conçu par John Von Neumann. Bien entendu, on disposait déjà de machines électromécaniques pour automatiser certains traitements simples de l'information comme le classement ou l'arithmétique, mais ces machines avaient toujours une structure simple et rigide. Au début des années cinquante, l'utilisation de l'électronique digitale naissante, associée au formalisme de la logique booléenne, ont apporté la puissance expressive et la souplesse nécessaires à la définition des composants de base des ordinateurs : portes logiques, bascules, éléments de mémorisation etc.

Au début des années soixante, le formidable élan de la technologie électronique, fondé sur les transistors et le silicium, a permis de réaliser des composants de base de plus en plus performants. Ils sont devenus plus compacts, plus fiables, plus rapides, et surtout beaucoup moins chers. Cependant, les composants des ordinateurs que nous connaissons effectuent les mêmes fonctions que ceux qui constituaient le premier ordinateur de Von Neumann. Ce qui différencie fondamentalement les ordinateurs, c'est leur architecture, c'est-à-dire la manière dont sont agencés les composants de base. Or, il s'est avéré que ces composants possèdent la propriété importante de pouvoir être facilement combinés entre eux, avec une grande souplesse. Il en a résulté un foisonnement d'architectures d'ordinateurs répondant à des besoins de plus en plus diversifiés.

Ceci a permis d'étendre le domaine d'application des ordinateurs, initialement dédié au calcul numérique et balistique, à ▶

presque toutes les activités de la société. On leur a demandé de faire des choses de plus en plus compliquées et ils sont devenus de plus en plus compliqués. Alors que les premiers ordinateurs étaient cantonnés dans les salles de calcul, aujourd'hui ils sont distribués partout : ils servent au contrôle des usines, des engins, des systèmes de régulations de la société (réseaux électriques, trafic…) etc. Pour cela, il est très souvent nécessaire qu'ils travaillent en temps réel avec le système auquel ils sont couplés. Ceci a donné naissance aux systèmes d'exploitation qui furent d'abord de simples outils logiciels chargés de gérer les programmes s'exécutant sur l'ordinateur.

Mais très vite, les systèmes d'exploitation ont dû incorporer des fonctions de service de plus en plus sophistiquées : gestion de programmes sur plusieurs machines, gestion d'applications distribuées en réseaux locaux ou bien au niveau d'une nation, gestion des contraintes temps réel dans l'industrie, gestion de bases de données et des banques d'informations.

En même temps, c'est la qualité des services demandés aux systèmes informatiques qui a évolué. Certes, on a demandé de nouvelles fonctions, mais surtout on a demandé un meilleur service.

Ceci se traduit par une demande accrue en performance de calcul (machines parallèles), en taille des données traitées (systèmes de gestion de bases de données) et surtout, de nouveaux concepts sont devenus prédominants comme la tolérance aux fautes et la sécurité des traitements.

Il résulte de cette évolution que dans l'architecture d'un ordinateur moderne, la part de l'effort en conception et en réalisation du matériel est souvent plus faible que la part de l'effort en conception et en réalisation des outils logiciels de base. Les coûts des investissements en logiciel étant très importants, leur évolution est comparativement lente devant l'évolution des structures matérielles sous-jacentes qui profitent, sans que cela ait de cesse depuis trente ans, des progrès exponentiels de la technologie du silicium.

Depuis une dizaine d'années, la technologie d'intégration des circuits permet de mettre sur une seule puce de silicium un très grand nombre de composants de base. On parle alors de technologie VLSI *(Very Large Scale Integration)* : on dépasse fréquemment le million de transistors et on annonce dix millions de transistors sur une seule puce en 1995. C'est ainsi que sont nés les microprocesseurs qui équipent les ordinateurs individuels actuels. Leur architecture est devenue de plus en plus complexe ; ce sont des machines à jeu d'instruction complexe ou CISC *(Complex Instruction Set Computers)*.

Pour maîtriser cette complexité, les informaticiens ont dû développer des outils logiciels pour automatiser au maximum les tâches répétitives intervenant dans la conception des puces, donnant naissance aux techniques de CAO (conception assistée par ordinateur) de circuits VLSI et, depuis peu, aux techniques de CAO d'architectures qui visent, à terme, l'automatisation complète de la conception et de la réalisation de systèmes matériels.

Malgré l'aide apportée par les outils de conception, les architectes ont rencontré des difficultés dans la mise en œuvre des machines de type CISC : elles sont devenues chères à produire (puces comportant des centaines de milliers de transistors), difficiles à programmer (réalisation de compilateurs complexes et lents), et leurs performances ne suivent plus la courbe des progrès de la technologie des circuits de silicium. C'est pourquoi, une nouvelle génération d'architectes a proposé un retour à un modèle de machine séquentielle "pure" dont le jeu d'instruction serait réduit au maximum : les machines à jeu d'instruction réduit ou RISC *(Reduced Instruction Set Computers)*. Ces nouveaux microprocesseurs permettent de réaliser depuis quelques années des stations de travail professionnelles extrêmement performantes.

Le concept d'architecture RISC semble bien être le dernier avatar du modèle de Von Neumann. C'est pourquoi, au début des années 90, la part essentielle de l'effort de recherche et de développement porte sur les nouvelles machines parallèles. Alors que les premiers supercalculateurs (multiprocesseurs et calculateurs vectoriels) n'étaient installés que dans quelques centres de calcul privilégiés du fait de leur coût exorbitant, la disponibilité de ressources matérielles abondantes et peu coûteuses permet d'envisager la démocratisation des machines parallèles. Ces nouvelles machines très parallèles atteignent déjà des performances impressionnantes et offrent la perspective d'une puissance gigantesque, dépassant de plusieurs ordres de grandeur celle des ordinateurs que nous connaissons.

Jean-Paul Sansonnet, directeur de recherche au CNRS.

MACHINES TRÈS PARALLÈLES

Les ordinateurs classiques fonctionnent de manière séquentielle. Mais il est possible d'atteindre des puissances de calcul beaucoup plus grandes avec des machines dites parallèles dont il existe une assez grande variété.

HIGHLY PARALLEL MACHINES - Parallel machines provide far greater computing power than conventional sequential machines. There are four classes of parallel machines: distributed systems, vector computers, superscalar microprocessors, dedicated operators. The programming for them is shared between three schools of thought. They do not actually compete as each is adapted to a specific range of applications.

Michel Cosnard
Jean-Paul Sansonnet

Bien que le modèle séquentiel ait joué un rôle prépondérant dans l'histoire des machines informatiques, très tôt les architectes ont cherché à améliorer les performances des ordinateurs. Von Neumann lui-même a proposé, dès le début des années cinquante, d'autres architectures fondées sur le concept d'automate cellulaire, malheureusement difficiles à réaliser avec la technologie dont on disposait à cette époque. Dans les années soixante, les ressources matérielles étant plus compactes et moins onéreuses, il a été possible d'envisager sérieusement la construction d'ordinateurs possédant des éléments répliqués travaillant en parallèle : bancs mémoire, unités de calcul, unités de contrôle, unités d'entrées/sorties etc. En 1969, M. J. Flynn a proposé un système de classification de ces architectures qui est toujours en vigueur malgré la venue de nouveaux modèles d'architectures parallèles (Fig. 1).

En quatre décennies, l'architecture des ordinateurs a été marquée par quatre grandes périodes (Fig. 2), mais c'est au début des années 80 seulement qu'on a commencé à voir fleurir les études académiques sur les architectures de machines massivement parallèles, études suivies souvent de très près (trop près ?) par des réalisations de prototypes industriels qui ont permis aux pionniers de la program- ▶

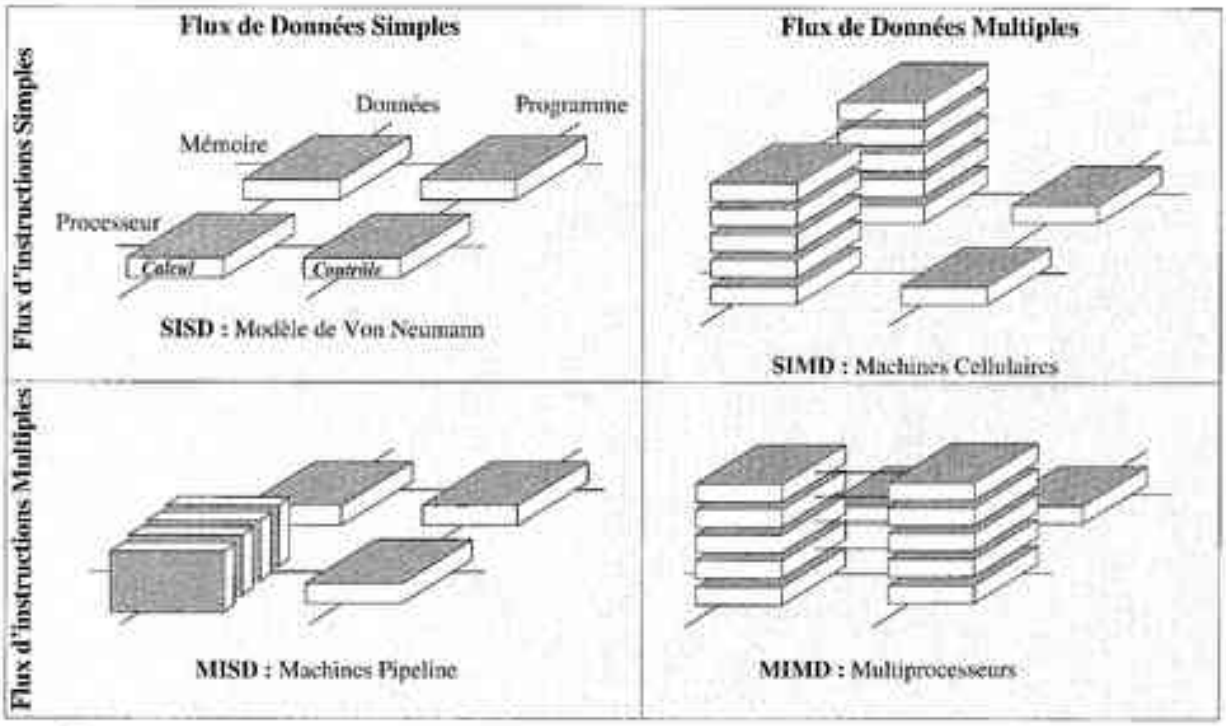


Fig. 1 - La classification historique de M.J. Flynn est fondée sur deux critères très simples : la machine a-t-elle un (Single Data Stream) ou plusieurs (Multiple Data Stream) flux de données ? La machine a-t-elle un (Single Instruction Stream) ou plusieurs (Multiple Instruction Stream) flux d'instructions ? Cette taxinomie fait ressortir les principales classes d'architectures actuelles : le modèle séquentiel courant dans les stations de travail et les ordinateurs personnels ; le modèle pipeline que l'on rencontre dans les supercalculateurs vectoriels pour le calcul scientifique où l'unité de calcul est constituée d'étages effectuant des opérations à la chaîne ; les machines cellulaires dont les applications sont très importantes en traitement d'image, où les unités de mémoire et les unités de calcul sont répliquées et exécutent toutes la même instruction en parallèle ; enfin, les systèmes multiprocesseurs que l'on rencontre dans les chaînes de contrôle/commande industrielles ou encore sous la forme de multi-calculateurs organisés en réseaux, où les unités exécutent leur propre programme en concurrence.

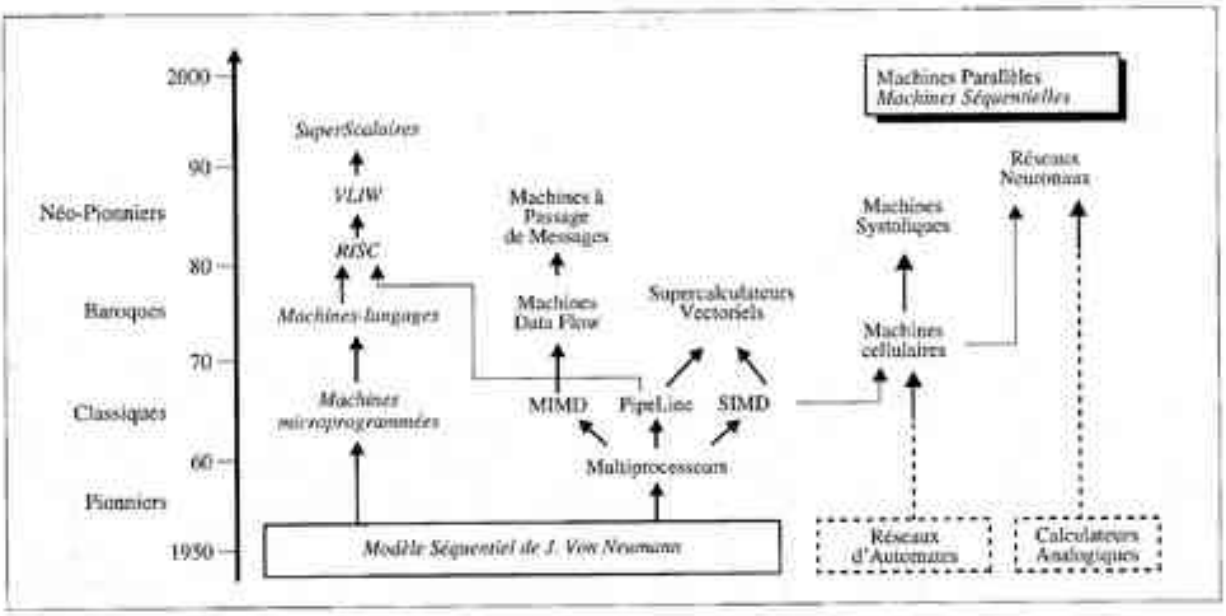


Fig. 2 - Evolution des concepts architecturaux. L'architecture des ordinateurs est marquée par quatre grandes périodes correspondant à peu de choses près aux quatre dernières décennies : l'ère des pionniers des années 50 ; l'ère classique des années 60 où le modèle séquentiel s'est stabilisé et s'est très largement répandu ; l'ère baroque* des années 70 où les architectes ont essayé de pousser à son extrême la sophistication du modèle de Von Neumann,et ont abouti à des architectures très complexes ; enfin, l'ère des néo-pionniers qui a commencé au début des années 80, essaye de défier de nouvelles bases avec le concept RISC et surtout est marquée par un trait commun : la généralisation des machines massivement parallèles.

mation parallèle de s'exercer, de développer les outils de base et maintenant de fournir les premières applications parallèles intégrées. En dix ans, les concepteurs de machines ont pu proposer beaucoup de modèles d'architectures parallèles nouvelles qui aujourd'hui arrivent à la maturité industrielle. On peut distinguer quatre classes correspondant à l'utilisation du parallélisme à des niveaux hiérarchiques différents :
- niveau système dans les systèmes distribués (exemple : les systèmes à Transputer) ;
- niveau machine dans les machines vectorielles (les machines CRAY), ou data-parallèles (*Connection-Machine* 5 de TMC) ;
- niveau microprocesseur dans les superordinateurs scalaires à mémoire partagée (microprocesseur ALPHA de DEC) ;
- niveau opérateur dans les circuits spécialisés (circuits ASICS de type XILINX).

Il n'y a pas réellement de concurrence entre ces différentes classes car elles répondent à des besoins très différents. Par contre, pour une classe donnée, il peut subsister des incertitudes quant au modèle qui l'emportera dans le futur proche. Par exemple, pour le niveau machine, entre le modèle vectoriel et le modèle data-parallèle il y a de nombreuses discussions pour savoir celui qui supplantera l'autre ; aujourd'hui le parc installé est celui des machines vectorielles, mais les machines data-parallèles commencent à se diffuser.

Trois écoles de programmation

Comme pour les classes d'architectures, on assiste à une stabilisation des concepts pour les modèles de calcul parallèle autour de trois modes de pensée attracteurs qui correspondent, de fait, à trois écoles :
- extraction automatique du parallélisme : en partant d'un programme déjà écrit dans un mode séquentiel, on utilise des outils logiciels d'extraction automatique pour générer les entités parallèles (exemple : les compilateurs FORTRAN pour les machines vectorielles ou les superordinateurs scalaires dotés d'un pipe-line),
- expression en terme de processus concurrents : le programmeur choisit une une structure parallèle dite à gros grain (par opposition au parallélisme à petit grain des machines pipe-line ou vectorielles) pour son programme et il doit se préoccuper lui-même des interactions entre les processus concurrents, comme les échanges de données et les synchronisations (langages concurrents pour machines MIMD),
- invention de nouveaux algorithmes (algorithmique dite data-parallèle), en partant de l'exploitation en parallèle des structures de données répétitives de l'application (langages *LISP - prononcer STARLISP, *FORTRAN pour machines SIMD/SPMD telles que la *Connection-Machine*).

Il n'y a pas réellement de concurrence entre ces trois écoles, car elles s'adressent à des classes de problèmes différentes. On aura besoin des techniques d'extraction automatique du parallélisme pour générer le code machine des nouvelles stations de travail qui utiliseront les nouveaux microprocesseurs superscalaires dont la caractéristique essentielle est d'exploiter le micro-parallélisme entre les instructions. Pour les applications industrielles générales (communication, contrôle/commande...), les systèmes distribués, fondés sur les langages à processus concurrents, et les machines à "passages de messages" qui communiquent de manière asynchrone via un réseau, sont appelés à se développer de plus en plus. Enfin, pour les applications intensivement parallèles, les machines data-parallèles sont indispensables. Il est probable qu'on verra coexister et coopérer des stations de travail, des systèmes concurrents et des machines data-parallèles à l'intérieur d'un même système informatique.

Michel Cosnard, professeur à l'Ecole normale supérieure de Lyon, LIP-IMAG (URA 1398 CNRS), ENSL, 46, allée d'Italie, 69364 Lyon Cedex 07.

Jean-Paul Sansonnet, directeur de recherche au CNRS, Laboratoire de recherche en informatique (URA 410 CNRS), Université de Paris-Sud, Bât. 490, 91405 Orsay Cedex.

LES DÉFIS DU CALCUL À TRÈS HAUTE PERFORMANCE

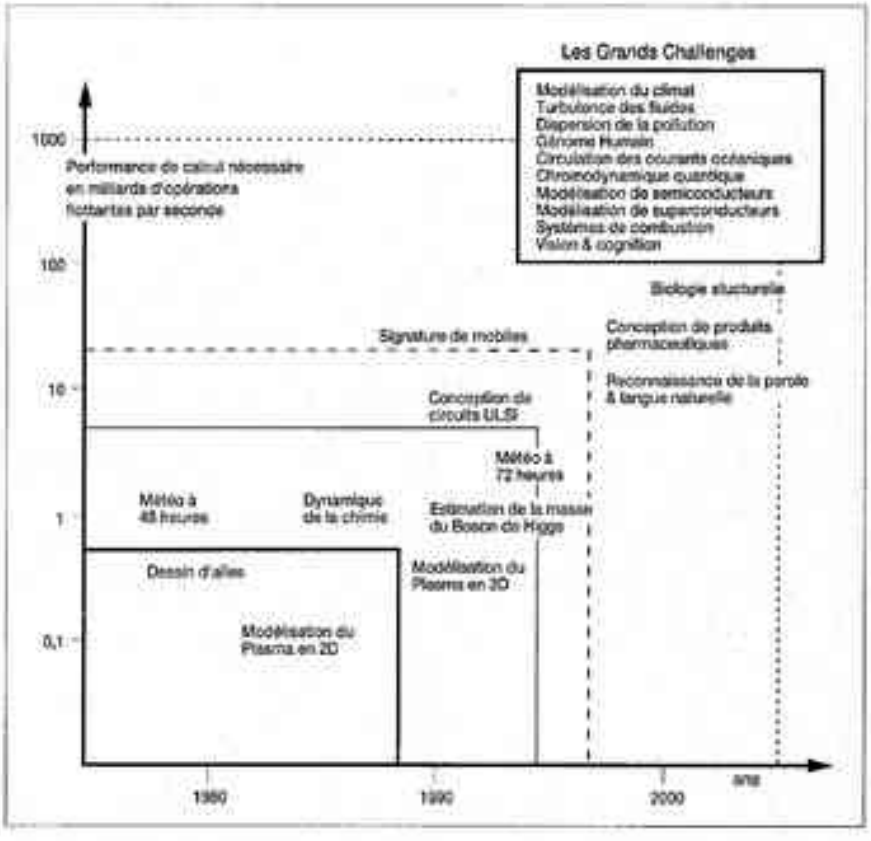
En 1991, aux États-Unis, D. Allan Bromley, conseiller pour les affaires scientifiques à la Maison Blanche, a lancé un programme très important pour accélérer le développement des technologies du calcul parallèle et des communications dans l'industrie américaine à l'horizon 2000. Il s'agit du programme HPCC (Highly Parallel Computing and Communication) pour lequel le document de présentation fait état d'un financement possible de 638 millions de dollars pour la seule année 1992, ce qui représente un accroissement de plus de 30 % des fonds accordés en 1991 à ce secteur.

Le programme HPCC est conduit par l'idée qu'il va falloir mettre en place des systèmes informatiques capables de délivrer une performance de calcul sans commune mesure avec ce qu'on a connu pour pouvoir faire face aux nouveaux besoins émergeant dans les applications de l'informatique (voir figure). Pour les USA, il s'est agi de définir un certain nombre d'applications scientifiques et industrielles désormais considérées comme "vitales pour les intérêts de la nation américaine".

Le programme HPCC comporte trois volets : une partie éducation/apprentissage, une partie recherche, une partie applications industrielles. Il est important de constater que, contrairement à des actions précédentes de même envergure, les aspects éducation et recherche jouent un rôle important car le passage d'un mode de pensée de type séquentiel à un mode de pensée de type parallèle va révolutionner les mentalités. Il faut donc mettre en œuvre une politique d'éducation et d'entraînement à l'utilisation de ces nouvelles machines si on veut qu'un véritable marché se développe (on sait qu'aujourd'hui les produits parallèles sont diffusés dans une communauté très restreinte).

Le programme est décomposé en quatre grands axes correspondant à quatre technologies pour le calcul parallèle :
- sytèmes de calcul à hautes performances
- logiciels et algorithmes avancés
- réseau national (USA) pour la recherche et l'éducation
- action et recherche de base, éducation et entraînement.

Au niveau européen, la commission de Bruxelles n'a pas tardé à réagir et, à son tour, a mis en place un groupe de réflexion en mai 1991 qui a émis un rapport à l'automne définissant une proposition de programme européen sur le même thème : HPC. Ce programme européen est en cours de spécification et il devrait déboucher sur un appel d'offres européen au cours de l'année.



Domaine d'application du "Grand Challenge". Au cours des dernières années, on a commencé à implémenter certaines applications intensivement parallèles, ou au moins une partie d'entre elles comme la météo qui fonctionne déjà très bien jusqu'à 48 heures. Certaines applications désignées dans le rectangle des "grands challenges" ne pourront voir une réelle mise en œuvre que si l'on est capable de leur fournir une puissance de calcul très importante.

LA CONCEPTION DES MACHINES ET DES CIRCUITS

Les circuits électroniques ne doivent pas seulement remplir leur rôle. Ils doivent aussi être facilement testables pour permettre d'éliminer ceux qui présentent des défauts.

CIRCUIT ENGINEERING - Computer aided engineering of architectures is a difficult but essential exercise. Using simulation, circuits can be developed more quickly, the validity of the architecture considered can be checked and operating tests devised.

Patrice Quinton

La réalisation de circuits intégrés et, plus généralement, des machines informatiques fait appel à une ingénierie très complexe. Celle-ci dépend des objectifs visés : machines programmables – microprocesseurs, par exemple – ou systèmes spécialisés. Parmi ces derniers, les ASICs (*Application Specific Integrated Circuits*) que l'on rencontre dans toutes les applications grand public, sont conçus pour exécuter un algorithme unique ou remplacer un grand nombre de composants électroniques. Les ASICs permettent souvent de diminuer le coût d'un système et d'augmenter sa fiabilité.

La conception des circuits impose trois contraintes :

- Le résultat final doit être absolument correct. A l'inverse d'un logiciel, un circuit intégré ne se corrige pas.
- Le circuit doit être conçu pour occuper le moins de surface possible et sa consommation électrique doit correspondre aux performances attendues.
- Enfin, le circuit doit être testable. Les rendements de fabrication varient de 20 % à 50 %. Il est impératif de discriminer avec certitude les bons circuits des mauvais.

L'idéal serait de pouvoir synthétiser automatiquement une architecture à partir d'une description des fonctions qu'elle doit remplir. Ceci permettrait une conception plus rapide, ce qui est un important avantage économique. La synthèse architecturale s'applique à tous les types de circuits spécialisés ou non : la structure d'un processeur programmable peut être déduite d'une description du jeu d'instructions qu'il doit exécuter, comme celle d'une unité logique des équations booléennes qui la gouvernent. Dans tous les cas, la synthèse doit aussi prouver la validité de l'architecture envisagée.

Valider le circuit

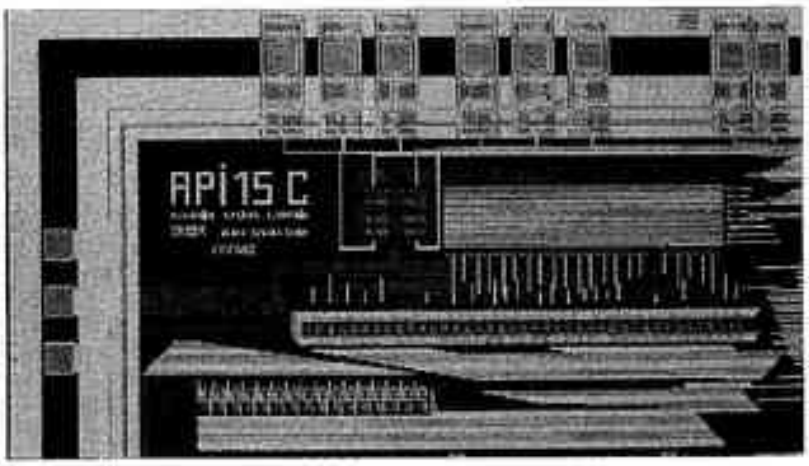
La méthode la plus utilisée pour valider un circuit est la simulation. C'est là une entreprise complexe, car les phénomènes à simuler ne relèvent pas tous du même niveau de modélisation. Dans l'absolu, il serait souhaitable de partir du modèle physique du circuit. Mais cette technique est, en pratique, inutilisable car trop longue. On ne l'emploie que pour vérifier le fonctionnement de certains éléments critiques.

La simulation n'est pas totalement satisfaisante. Pour prouver que le fonctionnement est parfaitement correct et qu'il correspond bien aux spécifications, il faudrait simuler le comportement dans tous les cas possibles, ce qui n'est pas réaliste. Aussi les techniques de preuve telles qu'elles existent pour les logiciels sont-elles de plus en plus fréquemment utilisées.

Permettre la conception d'architectures testables est un objectif majeur des recherches méthodologiques actuelles et ce, pour deux raisons. D'abord, il faut être capable, à l'issue de la fabrication, de séparer les composants corrects de ceux qui ne le sont pas. Le test utilisé doit être rapide, car les machines de test sont très coûteuses. Ensuite, il faut pouvoir détecter facilement des pannes éventuelles et diagnostiquer leur cause. La testabilité d'un circuit suppose la définition a priori d'un modèle des fautes que l'on souhaite détecter. Il faut ensuite s'assurer qu'elles pourraient être produites par des configurations d'entrées bien choisies et surtout être sûr qu'elles se traduisent par un signal repérable sur les circuits de sortie.

Enfin, les méthodes de conception – souvent très complexes – doivent pouvoir se concrétiser sous la forme d'un logiciel facile d'emploi. Une part importante de la recherche est donc consacrée à la mise au point d'algorithmes de faible complexité, capables d'exécuter rapidement les tâches de vérification ou de synthèse, et de concevoir les logiciels correspondants.

Patrice Quinton, directeur de recherche au CNRS, IRISA, Campus de Beaulieu, 35042 Rennes Cedex.



Circuit visualisé avec les outils de CAO Solo 2000 de ES2 : un fragment de circuit intégré visualisé sur des outils de conception assistée par ordinateur. La conception d'architecture se fait avec l'aide de programmes de conception assistée extrêmement complexes, qui permettent d'examiner le circuit à plusieurs niveaux d'abstraction, et de vérifier que ces représentations sont cohérentes (Photo INRIA - A. Eidelman).

LES ARCHITECTURES SYSTOLIQUES

L'étude d'architectures nouvelles dont la structure facilite la conception est une voie de recherche intéressante. Les architectures systoliques en sont un exemple. Considérons le produit de convolution que l'on retrouve dans un très grand nombre d'applications où des systèmes spécialisés sont nécessaires : étant donné une suite de valeurs $(x_i)_{i \geq 0}$, il s'agit de calculer les valeurs $y_i = \sum_{k=0}^{n} w_k x_{i-k}$, les coefficients w_k étant donnés. Une machine systolique permet d'effectuer le calcul d'une convolution de façon très efficace, à la volée. Cette machine est construite à partir de "cellules" élémentaires représentées sur la figure a.

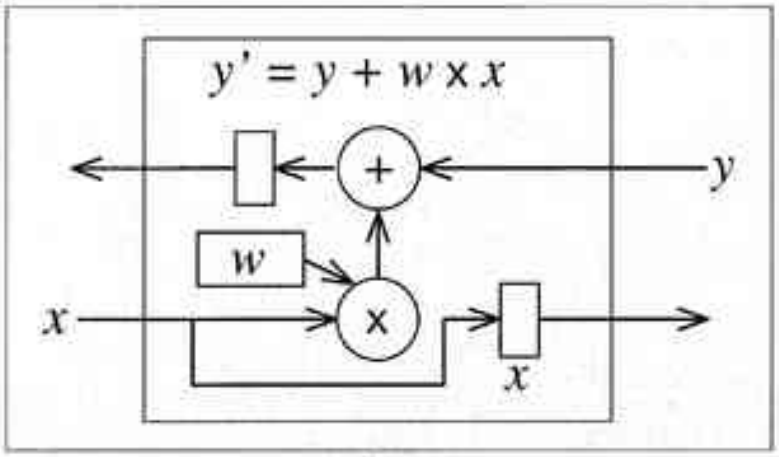


Fig. a

Une cellule reçoit en entrée une valeur x (en bas à gauche) et une valeur y (en haut à droite), et contient dans une mémoire un coefficient w. Elle calcule un résultat $y' = y + w \times x$, envoyé à gauche, et transmet x inchangé vers la droite. Ces résultats sont mémorisés dans des registres, pour pouvoir être utilisés au cycle suivant par les cellules voisines. En connectant trois cellules de ce type, chacune mémorisant respectivement w_0, w_1 et w_2, on réalise une machine parallèle qui peut calculer à chaque instant un résultat y_i.

La figure b illustre le fonctionnement de cette architecture sur le calcul de $y_2 = w_0 x_2 + w_1 x_1 + w_2 x_0$.

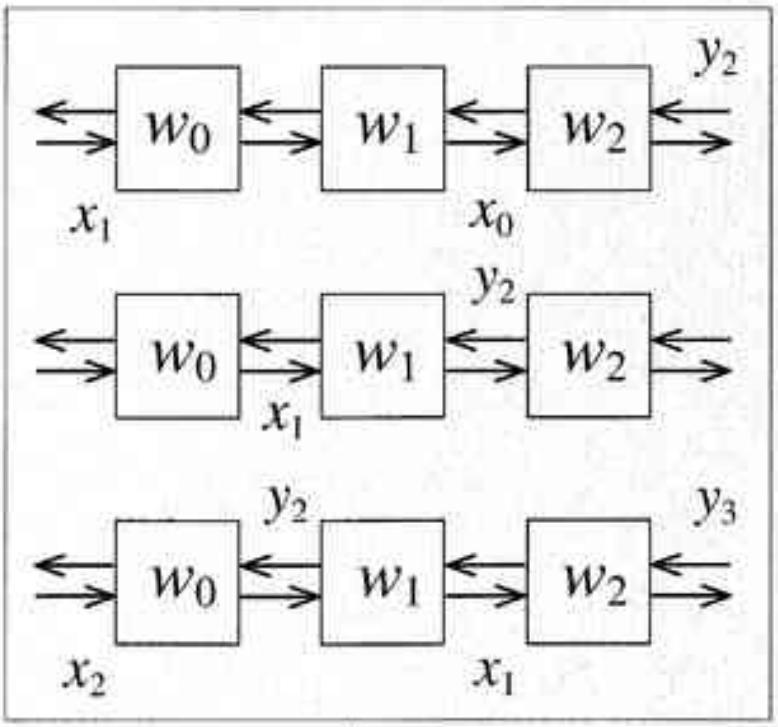


Fig. b

Les coefficients x entrent dans la machine par la cellule de gauche, tous les deux cycles. On entre des y initialisés à 0 dans la cellule de droite, et les résultats cherchés y_2, y_3, etc. sortent de la cellule de gauche à raison d'un résultat tous les deux cycles.

Les architectures systoliques présentent l'intérêt d'être très modulaires, et donc particulièrement faciles à intégrer. En effet, la réalisation d'un circuit pour la convolution d'ordre n se déduit du circuit d'ordre $n - 1$ par simple adjonction d'une cellule. De plus, les cellules sont connectées de façon locale, ce qui évite les problèmes électriques posés par les interconnexions longues.

On trouve de telles architectures dans des dispositifs de traitement de signal (radar, sonar, compression d'image, traitement de données).

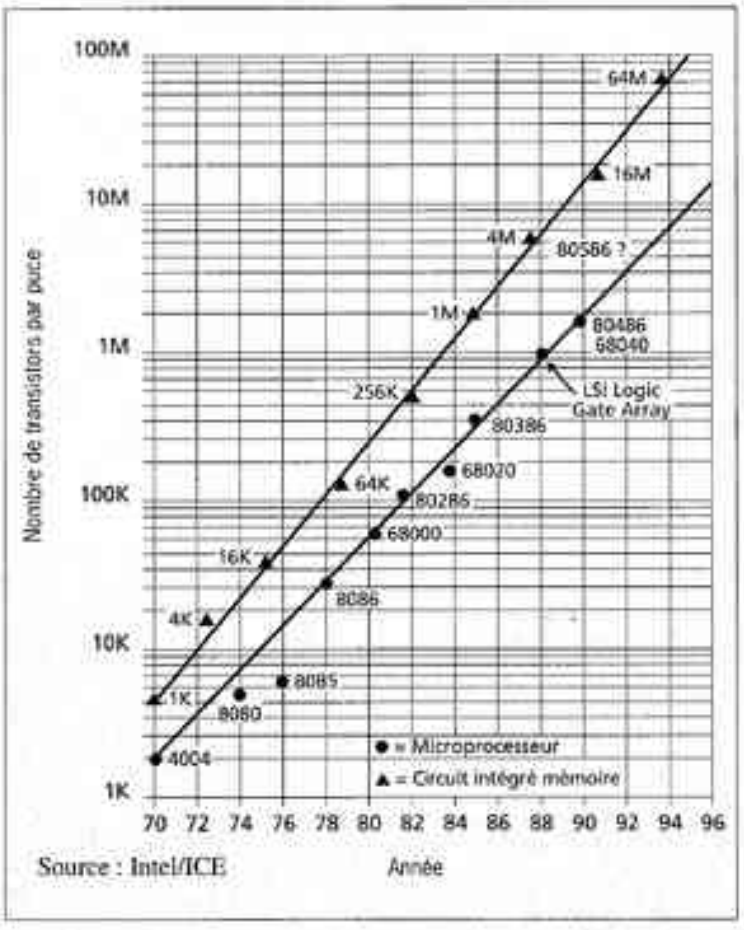
LA CROISSANCE EXPONENTIELLE DES CIRCUITS INTÉGRÉS

De 1985 à 1990, le coût de l'unité de puissance (MIPS) des stations de travail a été divisé par dix puisque le prix du MIPS de microprocesseur a baissé dans un rapport trois à quatre, et celui du bit de mémoire a été divisé par près de dix alors que son temps d'accès était réduit de moitié... Les technologies les plus avancées aujourd'hui réalisent des transistors dont les motifs ont une taille qui approche 0,5 microns et qui fournissent trois niveaux d'interconnexion métallique.

Ces progrès des technologies de fabrication ont pour conséquence une croissance régulière de la complexité (mesurée en nombre de transistors) des circuits intégrés, représentée sur la figure.

Cette croissance est exponentielle comme l'indique l'échelle verticale qui est logarithmique : la complexité des circuits de type mémoire dynamique (DRAM) augmente de 50 % par an, celle des microprocesseurs de plus de 30 %. Ceci résulte de l'amélioration de la finesse des géométries, mais aussi de l'augmentation de la surface des circuits, qui est de 13 % par an. Ces évolutions trouveront certainement leur limite ; mais les spécialistes s'accordent à penser que, pour ce qui est de la finesse des géométries, la limite se situera en dessous de 0,1 micron, et que les courbes présentées peuvent être extrapolées jusqu'en 1996, voire jusqu'à la fin de la décennie.

L'informatique, si elle bénéficie de cette évolution, y joue aussi un rôle actif. Par exemple, une part croissante du marché des semiconducteurs concerne les circuits conçus spécifiquement pour une application, ou ASICs. Mais concevoir un circuit de 100 000 ou 1 000 000 de transistors est une entreprise délicate et périlleuse, la moindre erreur rendant le circuit inopérant, et coûtant, outre le prix de la fabrication, plusieurs mois de délais. Pour faciliter cette tâche et la rendre plus sûre, il est indispensable d'élaborer des outils de CAO sophistiqués dont l'étape ultime sera peut-être le compilateur de silicium qui, après fourniture d'une description des spécifications du circuit dans un langage de haut niveau, sera capable de produire les dessins des masques et les séquences de test...



Guy Mazaré, professeur à l'Institut national polytechnique de Grenoble, Laboratoire de génie informatique - Unité de génie matériel, IMAG, (LGI-IMAG) (URA 398 CNRS), 46, avenue Félix Viallet, 38031 Grenoble Cedex.

GCIS, CMP ET EUROCHIP

Une recherche très active se développe dans ce domaine et les équipes françaises, fédérées par le GDR GCIS (Groupement circuits intégrés silicium) du CNRS y tiennent une place importante.

Pour permettre aux chercheurs d'expérimenter, c'est-à-dire de concevoir des circuits intégrés complexes et de les faire fabriquer, plusieurs actions se sont développées en France et en Europe. Le service CMP, assuré par le CNRS et les universités, se charge de rassembler les dessins des masques de plusieurs circuits différents, de faire fabriquer ces circuits par un industriel (le "fondeur"), de découper les tranches et de renvoyer les puces réalisées à leur concepteur; en Europe, Eurochip fédère les différents services de ce type, permettant des fréquences de fabrications plus élevées et des technologies plus variées.

LES ARCHITECTURES RISC

L'exploitation de la fantastique puissance de calcul des processeurs RISC exige une amélioration des hiérarchies mémoire et des procédés de communication.

Daniel Etiemble

RISC ARCHITECTURES - Improving performance of the memory hierarchy is compulsory to reduce the gap between peak and sustained performance of RISC processors. The impact of optimizing compilers is growing with the new generation of superscalar and superpipelined RISC microprocessors. Efficient memory hierarchies and interconnection networks are key points to use the computational capabilities of RISC processors in massively parallel architectures.

Les microprocesseurs RISC sont maintenant très largement utilisés, des stations de travail aux machines massivement parallèles. Si l'architecture RISC a été initialement caractérisée par un jeu d'instructions réduit, la véritable différence entre RISC et CISC est ailleurs. Alors que les CISC ont des instructions de longueur variable, dont la plupart accèdent à la mémoire, les RISC ont des instructions de longueur fixe et seules les instructions Load et Store accèdent à la mémoire. Ces caractéristiques permettent de pipeliner très efficacement l'exécution des instructions, et de réduire à la fois le nombre moyen de cycles par instruction (CPI) et le temps de cycle (Tc).

Le CPI est de l'ordre de 6 à 12 pour le VAX 6000 (CISC typique) et inférieur à 1,5 pour la première génération des RISC. Le temps de cycle, qui décroît exponentiellement avec les années, est descendu à 5 ns pour le processeur 21064 ALPHA de DEC, avec une technologie CMOS 0,75 µm.

Performances maximales et performances réelles

La puissance crête, exprimée en MIPS (Millions d'Instructions par Seconde) est inversement proportionnelle au produit CPI × Tc. Malheureusement, l'exigence contradictoire d'un temps de cycle court et d'une grosse capacité mémoire impose une hiérarchie mémoire, avec caches et mémoire principale, qui diminue les performances réelles.

L'évolution technologique accentue l'écart entre les temps de cycle des caches et de la mémoire principale. Pour éviter un écart croissant entre performances maximales et performances effectives des machines, il faut diminuer à la fois le nombre de défauts de cache et leur temps de traitement. Les défauts de cache, même à des taux très faibles, provoquent des débits de transfert extrêmement élevés sur les bus système, et encore plus dans le contexte multiprocesseurs. De nouvelles techniques de transmission, comme les liaisons série haut débit compatibles avec les technologies silicium, peuvent ouvrir de nouvelles possibilités, ou s'avérer indispensables.

Microparallélisme et compilation

Pour exécuter une instruction par cycle avec les RISC de première génération, le rôle des compilateurs optimisants était déjà décisif. En réordonnançant les instructions, l'optimisation peut diminuer le nombre d'accès mémoire, d'opérations et l'impact des branchements. Des améliorations de l'ordre de 20 à 60 % ont été citées sur la station DEC 3100 sur des programmes comme TeX, Spice et GCC.

En 1992, une nouvelle génération de RISC capable d'exécuter plusieurs instructions par cycle apparaît, comme le MS8110 qui utilise l'approche superscalaire (Fig. 1), le MIPS R4000 qui utilise l'approche superpipeline (Fig. 2), ou le 21064 de DEC qui utilise les deux techniques. Gérer l'exécution simultanée de plusieurs instructions accentue encore le rôle du compilateur, pour éviter les dépendances de données et diminuer l'impact des ruptures de séquence.

Si la réalisation des superscalaires et superpipelines semble maîtrisée, il reste des questions ouvertes. Le microparallélisme intrinsèque des programmes des stations de travail étant limité, de nouvelles approches sont envisagées pour utiliser au mieux le matériel, comme l'entrelacement de l'exécution de plusieurs processus.

Architectures massivement parallèles.

Aux problèmes de hiérarchie mémoire s'ajoute pour les machines massivement parallèles la réalisation du réseau d'interconnexion. Doter les processeurs RISC d'une puissance de communication à la hauteur de la puissance de calcul est le problème à résoudre maintenant, pour aller vers une nouvelle génération de machines massivement parallèles.

Daniel Etiemble, professeur à l'Université de Paris-Sud, directeur du GDR Architectures nouvelles de machines, Laboratoire de recherche en informatique (URA 410 CNRS), Université de Paris-Sud, Bât. 490, 91405 Orsay Cedex.

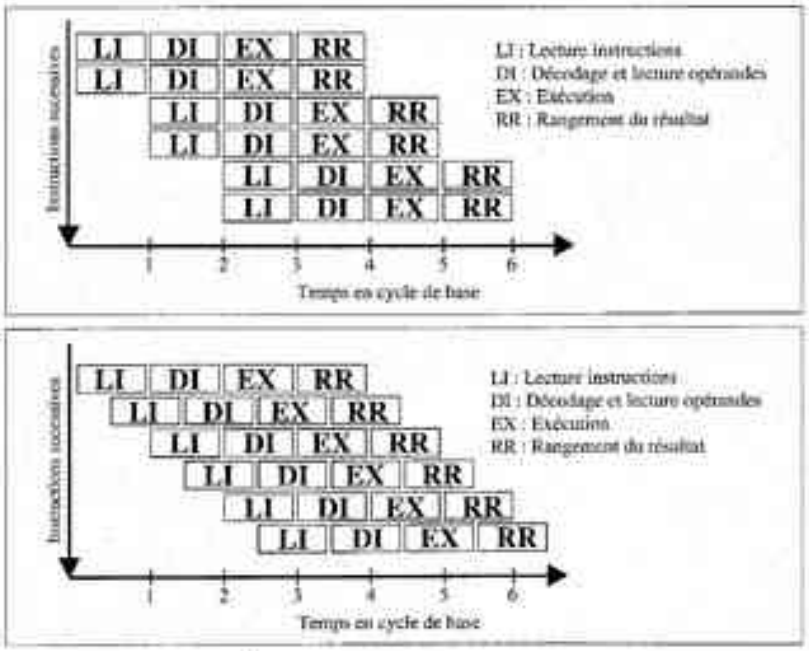


Fig. 1 - Pipeline superscalaire de degré 2.
Fig. 2 - Superpipeline de degré 2.

L'OPTIQUE ET L'ORDINATEUR

L'ordinateur optique sera-t-il demain un appareil courant ? Sera-t-il compatible avec les machines actuelles ou le parangon des futures machines parallèles ?

OPTICS AND COMPUTERS - The fully optical computer already exists. But its economic prospects are still uncertain. Will it enter the competitive arena of semi-conductor computers or come into its own on a quite different plane?

Pierre Chavel
Jean Taboury

L'ordinateur optique au sens strict a longtemps été un mythe : il est maintenant une réalité, puisque la petite entreprise américaine Opticomp a fabriqué une telle machine, baptisée DOC 2. Pour l'utilisateur, le recours à l'optique est transparent : DOC 2 possède le jeu d'instructions d'une station Sun Sparc 4/110. Pourtant, son unité centrale ne fait appel qu'à des composants optoélectroniques commerciaux : diodes lasers, cristaux acousto-optiques et photodiodes à avalanche. Cette machine résulte d'un travail de recherche et sa commercialisation n'est pas envisageable dans un avenir proche pour des raisons économiques, mais ses performances ne sont en rien méprisables puisque la fréquence d'horloge est de 100 MHz ; l'accès à la mémoire électronique qui stocke les codes d'instructions ralentit toutefois la cadence effective et il faudra l'apparition de nouvelles générations de composants optoélectroniques pour dépasser ce handicap.

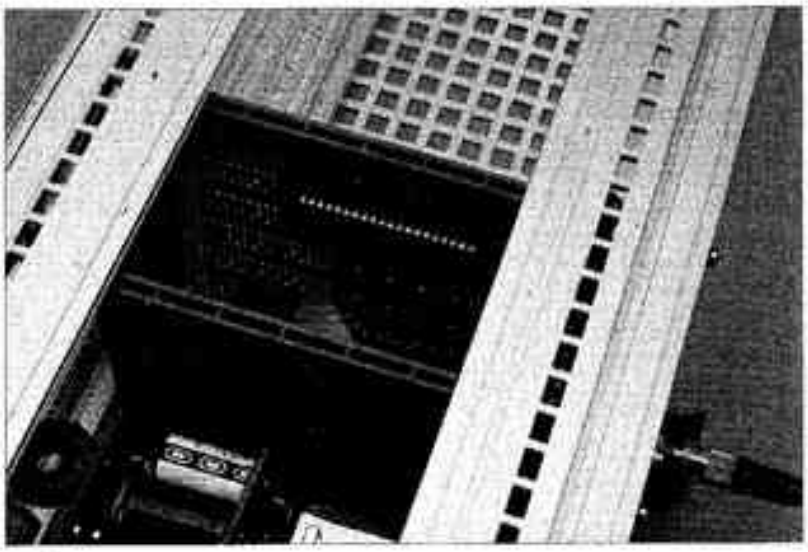
Mais précisément, l'intérêt principal de l'optique n'est pas la possibilité de reproduire un ordinateur existant ; il provient plutôt du fait que la technologie du silicium, omniprésente en informatique comme on le sait, essaime maintenant vers d'autres semiconducteurs, mieux adaptés que ce dernier aux fonctions optiques : c'est notamment le cas de l'arséniure de gallium et de ses frères de la famille appelée III-V, composés d'un élément de la troisième et d'un élément de la cinquième colonne du tableau de Mendeleiev. Le composant le mieux connu de la série est la diode laser, dont un exemplaire grand public, produit pour quelques francs, équipe chaque lecteur de disque laser, et dont le haut de gamme permet les télécommunications transocéaniques par câble optique. L'application de ces technologies aux ordinateurs requiert encore d'autres dispositifs qui sortent des laboratoires depuis peu : ainsi, ces deux dernières années, les laboratoires Bell ATT ont successivement annoncé une matrice de diodes lasers comprenant 2 millions de lasers intégrés sur une puce d'un centimètre carré, puis la commercialisation de la matrice de bistables optiques, présentée comme le prototype des composants pour ordinateurs à base de faisceaux lumineux. Les industriels européens et japonais ne sont pas à la traîne et développent d'autres composants, à base de semiconducteurs ou de cristaux liquides nouveaux.

Compatibles ou non ?

A quelles machines ces progrès ouvrent-ils la voie ? Deux écoles s'affrontent encore. D'un côté, la conception de DOC 2 repose sur le postulat qu'une machine optique doit assurer une compatibilité parfaite avec les générations de machines qu'elle aspire à surclasser : elle profitera ainsi des marchés existants, de la compétence acquise par les utilisateurs et des outils logiciels disponibles ; mais le but reste lointain. Pour l'atteindre, il faudra repenser toute l'architecture de façon à profiter au maximum de l'attrait le plus évident et le plus intuitif de l'optique : la propriété des faisceaux lumineux de se propager en grand nombre à travers l'espace libre sans interagir, autrement dit la capacité de la lumière à réaliser des interconnexions denses et sans diaphonie. Dans ce cas, pourquoi ne pas recourir à l'optique de préférence dans les machines où le besoin d'interconnexions est le plus grand : les machines parallèles ? Telle est l'approche retenue par la deuxième école qui mise sur l'optique essentiellement pour augmenter le parallélisme, soit sous la forme de connexions entre composants électroniques (puces, cartes, fonds de panier), soit au sein de processeurs analogiques spécialisés dans certaines opérations importantes de traitement du signal telles que la détection des contours pour la vision par ordinateur.

Pierre Chavel, directeur de recherche au CNRS, Institut d'optique théorique et appliquée (URA 14 CNRS).

Jean Taboury, professeur à l'Université de Paris-Sud, Institut d'optique théorique et appliquée (URA 14 CNRS), Université Paris-Sud, BP 147, 91403 Orsay Cedex.



Générateur optique de tableaux de nombres aléatoires parallèles, P. Lalanne et J.-C. Rodier, Institut d'optique, en collaboration avec l'Institut d'électronique fondamentale, Orsay, URA 22 CNRS.

LES SYSTÈMES D'EXPLOITATION

Sans eux l'informatique n'existerait pas. Ils gèrent les ressources de l'ordinateur, aussi bien matérielles que logicielles, et font d'une machine électronique un instrument utilisable par l'homme.

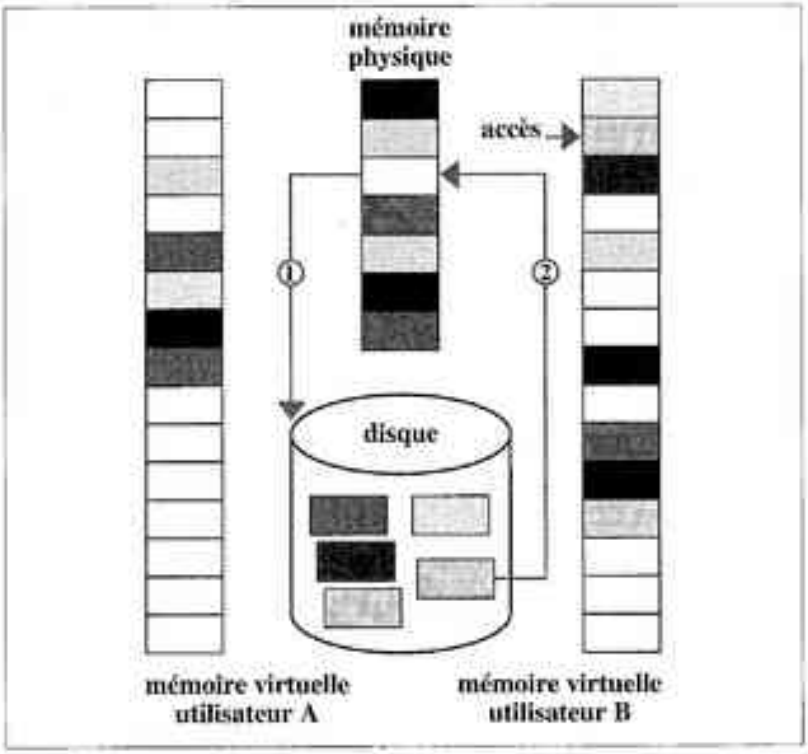
OPERATING SYSTEMS - They are at the heart of all systems. Designed to optimally handle the material resources of computers, operating systems have evolved becoming more user-friendly. To such an extent that the user now often faces a virtual machine that bears no relationship with the actual machine. Distributed processing raises major new challenges for operating systems.

Sacha Krakowiak

Le rôle du système d'exploitation d'un ordinateur est de gérer ses ressources matérielles et logicielles : il doit assurer que la bonne information arrive au bon endroit au bon moment et sous une forme intelligible, et que les processeurs, mémoires et organes de communication sont disponibles pour les tâches à assurer. Mais le rôle du système d'exploitation ne se limite pas à cette fonction de gérant de ressources. Il doit aussi masquer aux utilisateurs la complexité du fonctionnement interne de l'ordinateur, leur permettant ainsi de se concentrer sur le problème à résoudre en oubliant les détails de l'intendance. Le système d'exploitation donne à l'utilisateur la vision d'une "machine virtuelle", image idéalisée de la machine physique sous-jacente. La notion de ressource virtuelle permet de masquer aux utilisateurs la limitation (en taille et en nombre) des ressources physiques correspondantes. Elle permet aussi de dissimuler les détails de leur mise en œuvre, leur caractère hétérogène (variété des supports de mémoire, processeurs multiples), et même dans certains cas leurs défaillances. La commodité d'utilisation pour l'usager final se paie évidemment par la complexité du système lui-même. La tendance actuelle de diffusion des outils informatiques vers une communauté de plus en plus large d'usagers impose donc aux concepteurs de systèmes des exigences croissantes.

Deux moteurs : l'évolution des matériels et les besoins des utilisateurs

L'évolution des systèmes d'exploitation a subi une double influence : celle de la technologie du matériel et celle des besoins des utilisateurs. La conception des premiers systèmes d'exploitation a été dominée par le souci d'exploiter au mieux des ordinateurs centralisés coûteux. Les systèmes en temps partagé ont restitué aux utilisateurs, à la fin des années 60, des possibilités d'interaction directe. Les postes de travail individuels, apparus au début des années 80 (mais déjà expérimentés en laboratoire depuis 1975), ont apporté aux utilisateurs l'autonomie totale et des capacités importantes d'interaction, notamment via leurs interfaces graphiques. Les besoins de communication et de partage, ainsi que la limitation des ressources locales, ont ensuite imposé de connecter ces machines en réseau pour leur permettre de dialoguer entre elles et d'accéder à des serveurs spécialisés. La communication est d'abord apparue sous la forme d'opérations spécialisées ajoutées aux systèmes existants ; on assiste aujourd'hui à



La gestion de la mémoire. A un instant donné, les informations contenues dans les mémoires virtuelles sont réparties entre la mémoire physique et le disque. Cette répartition est invisible aux utilisateurs qui accèdent à leur mémoire virtuelle comme si elle était réelle. Les informations sont découpées en blocs de taille fixe appelés pages. Le système gère les tables qui permettent de connaître la localisation de chaque page ; lors d'un accès à une page située sur disque, il doit l'amener en mémoire physique, en évacuant au besoin sur disque une autre page inutilisée.
On peut remarquer que certaines pages figurent dans les deux mémoires virtuelles (bien que leur représentation physique reste unique) ; elles sont partagées entre les deux utilisateurs. Des travaux récents portent sur les mémoires virtuelles réparties, dont le support physique comporte plusieurs machines connectées par un réseau de communication.

une intégration beaucoup plus étroite, sous la forme de systèmes d'exploitation répartis.

Un système d'exploitation moderne est un programme complexe et de grande taille (des milliers de modules, des dizaines de mégaoctets). La structuration hiérarchique en couches permet de maîtriser cette complexité.

Le noyau du système regroupe les mécanismes chargés du contrôle direct des ressources centrales de l'ordinateur : gestion des processeurs (donc activation et synchronisation des processus), mécanismes d'interruption. On lui associe également le pilotage des organes d'entrée-sortie et des communications élémentaires.

Le gérant de mémoire (voir figure) commande les unités de contrôle d'accès à la mémoire et assure l'allocation des zones de mémoire physique; il gère les mécanismes de pagination qui permettent de réaliser des mémoires virtuelles pour les utilisateurs.

Le système de gestion de fichiers conserve les informations confiées au système par les usagers, ainsi que l'information interne au système, sous forme d'unités indépendantes appelées fichiers, stockées en mémoire secondaire. Un catalogage hiérarchique facilite le classement des fichiers et la recherche d'un fichier particulier.

Les modules d'administration du système, inaccessibles aux utilisateurs ordinaires, fournissent les outils nécessaires à la gestion des ressources (configuration, réglage), des usagers (droits d'accès), et des informations (placement et protection des fichiers).

Les programmes utilitaires, dont le nombre peut être considérable, fournissent des services divers qui utilisent les couches basses du système et contribuent à sa valeur ajoutée : communication, aide à la construction et à la mise au point de programmes, documentation en ligne, etc.

De l'empirisme à la doctrine

La période 1965-1980 a vu le domaine des systèmes d'exploitation passer d'un ensemble de techniques empiriques à un corps de doctrine bien organisé et maîtrisé, notamment avec l'émergence des notions de processus, de segment, puis d'objet.

La conception des systèmes d'exploitation répartis est un thème de recherche actif, tant sur le plan conceptuel que sur le plan pratique. Parmi les problèmes les plus difficiles figurent ceux posés par l'accroissement de la taille des réseaux (des milliers de stations). Cette croissance complique l'administration des systèmes et la maîtrise de leurs performances. Elle amplifie également les problèmes de dis-

LE SYSTÈME, OMNIPRÉSENT MAIS INVISIBLE

Je suis devant mon ordinateur personnel pour mettre au point la dernière version de cet article. L'article est représenté sur l'écran par un petit dessin symbolique (une icone). Je désigne l'icone avec la souris, "double clic" sur le bouton : le voyant lumineux du disque clignote, le texte s'affiche sur l'écran. Tout cela a duré moins de deux secondes, moins de temps qu'il n'en faut pour lire ces quelques lignes. Aucun calcul n'a été fait, aucune information n'a été créée pour mon compte; et pourtant, l'ordinateur a exécuté quelques dizaines de milliers d'instructions. Les programmes qui ont été mis en œuvre sont ceux du système d'exploitation.

Analysons ce qui s'est passé. L'événement déclenchant a été le "clic" de la souris, interprété par l'ordinateur comme une interruption, c'est-à-dire le lancement d'une séquence d'instructions prioritaires. La position de la souris a été détectée sur l'écran. Une table de correspondance a été consultée et l'icone a été reconnue comme celle d'un fichier de texte à traiter par un programme de traitement de texte. Le système a réservé de la place en mémoire centrale et est allé rechercher l'emplacement sur disque des deux fichiers : texte et traitement de texte. Il a déclenché leur transfert depuis le disque vers les places réservées en mémoire. Il a enfin lancé l'exécution. Un programme d'affichage a présenté sur l'écran l'image de la première page : le système a cédé la place au programme d'application, le vrai travail peut commencer.

Je suis maintenant devant ma station de travail et le texte est celui d'un ouvrage auquel je collabore avec plusieurs autres auteurs. Pour l'utilisateur, le déroulement des événements est analogue ; mais la partie cachée est nettement plus complexe. Les fichiers sont stockés sur une machine distante (un serveur), peut-être dans un autre bâtiment ou à l'autre bout de la ville ; le serveur partage son temps entre toute une communauté d'utilisateurs ; le texte du document doit être protégé contre les manipulations incohérentes (et les indiscrétions). L'opération a nécessité l'intervention de deux systèmes : celui du serveur, qui est allé chercher les informations sur son disque, a mis en œuvre les protections d'accès, a transmis les informations sur une ligne de communication (en traitant peut-être d'autres requêtes dans l'intervalle), et celui de la station de travail, qui a reçu les informations transmises et les a affichées dans l'une des multiples fenêtres qu'il gère sur son écran et qui me permettent par exemple de lire mon courrier ou d'insérer dans mon document des parties récupérées ailleurs. Des conventions communes ont permis à ces deux systèmes de dialoguer.

LES PRINCIPAUX SYSTÈMES D'EXPLOITATION

OS/360 : système d'exploitation des ordinateurs IBM de la série 360 (lancée en 1964). C'est l'un des programmes les plus complexes jamais construits, et sa maintenance (il était écrit en langage assembleur) posait des problèmes insurmontables. Il est néanmoins l'ancêtre des systèmes ultérieurs d'IBM, notamment les systèmes à mémoire virtuelle.

Multics : développé au Massachussetts Institute of Technology (MIT) à partir de 1965 (initialement en collaboration avec Bell Labs et General Electric), ce projet de système universel en temps partagé a été à l'origine de progrès conceptuels capitaux : mémoire virtuelle segmentée, système hiérarchique de gestion de fichiers, anneaux de protection. Bien que ses débuts aient été difficiles, il a donné naissance à un produit commercial distribué par Honeywell, puis par Bull.

UNIX : la première version d'UNIX a été développée en 1971-73, dans une quasi-clandestinité, par deux chercheurs de Bell Labs transfuges du projet Multics, et en réaction contre la lourdeur supposée de ce dernier. Rapidement adopté (et enrichi) par la communauté des chercheurs, ce système s'est ensuite imposé comme standard de fait sur les stations de travail, et son champ d'application continue de s'étendre. Il présente néanmoins des lacunes remontant à ses origines, par exemple dans le domaine de la protection.

MS/DOS : développé par Microsoft, ce système doit sa popularité au fait que son support est l'IBM PC (et ses compatibles), le micro-ordinateur le plus largement répandu. Il garde toutefois une conception ancienne, compensée pour l'utilisateur par des outils puissants de gestion d'interface.

Chorus : issu de travaux lancés par l'INRIA et poursuivis dans la société Chorus Systèmes depuis 1985, Chorus est un micro-noyau destiné à servir de support à des systèmes généraux (dont une version d'UNIX) et spécialisés (notamment dans le domaine du temps réel), sur des réseaux locaux et des multiprocesseurs.

Mach : Mach est un micro-noyau développé à l'Université Carnegie Mellon. Son champ d'application est analogue à celui de Chorus, mais, contrairement à ce dernier, il a été construit par transformation progressive d'un système UNIX complet.

ponibilité des services et de maintien de la cohérence des informations partagées.

L'interpénétration croissante de l'informatique et des communications conduit à construire des systèmes qui intègrent les communications multimédia : données textuelles, images, voix. Les développements dans ce domaine ne font que commencer; la gestion des communications multimédia impose aux systèmes des contraintes fortes de débit, de capacité de stockage et de temps de réponse.

L'importance des investissements en construction de logiciels et le renouvellement rapide du matériel entraîné par l'évolution technique rendent impérative la normalisation des systèmes ; il n'est plus concevable de réécrire les programmes d'application à chaque changement de matériel ou de système. On assiste aujourd'hui à un mouvement en faveur de systèmes dits ouverts, dont les principales interfaces sont définies de manière stable, les composants eux-mêmes pouvant être renouvelés et améliorés à condition de respecter les interfaces fixées.

Un aspect de cette évolution est l'apparition de micro-noyaux qui regroupent les fonctions élémentaires de gestion de processus, de mémoire et de communication dans un noyau de taille réduite sur lequel peuvent être construits, à la demande, des systèmes spécifiques. On espère ainsi améliorer la modularité des systèmes et leur capacité d'évolution, tout en conservant une interface normalisée pour les fonctions élémentaires. Plusieurs systèmes peuvent aussi coexister sur un même micro-noyau, ce qui facilite les échanges et les évolutions.

Sacha Krakowiak, professeur à l'Université Joseph Fourier de Grenoble, directeur adjoint de l'unité Bull-IMAG systèmes (UMR 122 CNRS-Bull), Bull IMAG, 2, avenue de Vignate, ZI de Mayencin, 38610 Gières.

LES SYSTÈMES TEMPS RÉEL

L'attribut temps réel (en abrégé TR) s'applique aux systèmes informatiques dont les spécifications font apparaître des contraintes temporelles qui doivent être rigoureusement satisfaites. On confond souvent le concept de temps réel avec les notions de traitement en ligne, de traitement rapide ou de temps de réponse court. Par exemple, la réservation de place (trains, avions) n'est pas à proprement parler une application TR, même si l'on suppose qu'il faut répondre rapidement aux demandes des clients. C'est une application en-ligne, caractérisée par des temps de réponse moyens raisonnables, sans garantie d'existence de délai fini utile borné supérieurement. A l'inverse, la détection d'un obstacle imprévu sur une voie ferrée doit être effectuée par un système TR (embarqué ou au sol) à une date impérative, fonction de la vitesse du train et de sa distance à l'obstacle. La borne supérieure du délai de réaction doit être absolument respectée.

On connaît aujourd'hui quelques méthodes, formalismes, algorithmes, outils permettant de dérouler rigoureusement la chaîne de développement (passer des spécifications à une réalisation validée) lorsque les applications visées sont simples (conduite d'ascenceurs par exemple) et lorsque les architectures informatiques sont équivalentes à des systèmes centralisés (multiprocesseur à mémoire partagée par exemple).

De la même façon, l'offre industrielle actuelle en matière de systèmes d'exploitation TR ou de moniteurs TR (VRTX, pSOS, LynxOs, RTU, RealiX ou les UNIX dits répartis, pour ne citer que quelques exemples) n'apporte pas de solutions complètement satisfaisantes ou généralisables.

La recherche s'oriente selon deux directions :

- l'approche TR "synchrone" où l'on part de modèles simples, pour lesquels des solutions exactes sont connues, et où l'on élimine progressivement certaines des hypothèses restrictives sous-jacentes (connaissances préalables complètes et exactes sur les durées des tâches, les délais de communication, absence de dépendances entre tâches et d'occurence de défaillances, architectures équivalentes à un monoprocesseur) ;
- l'approche TR "asynchrone" où l'on part de modèles à connaissances préalables incomplètes (architectures distribuées à délais variables, occurences non prévisibles de conflits entre tâches, de défaillances, induisant des durées de tâches variables, lois d'arrivées des requêtes quelconques) et où l'on recourt à des algorithmes exécutés en-ligne pour lesquels on prouve qu'ils possèdent des propriétés particulières (garantie de terminaison de tâches parallèles asynchrones en temps fini borné par exemple).

La principale difficulté tient au fait qu'un système temps réel doit avoir un comportement correct (et donc prévisible) même en cas de surchage ou de défaillance. Les cas de surcharge ou de défaillance ne pouvant être anticipés, la recherche de nouveaux algorithmes en-ligne revêt une importance particulière, notamment en systèmes répartis. Des concepts récents, comme ceux de taux de compétitivité et d'optimalité en présence de connaissances incomplètes, paraissent prometteurs.

L'informatique temps réel bénéficie aujourd'hui d'un engouement croissant car les bases théoriques des solutions commencent à exister et aussi parce que les progrès technologiques en matière de composants permettent l'emploi d'algorithmes et d'outils complexes dont le coût était jugé exhorbitant il y a encore peu de temps.

Gérard Le Lann, directeur de recherche à l'INRIA, BP 105, 78153 Le Chesnay Cedex.

LES SYSTÈMES POUR MACHINES PARALLÈLES

L'utilisation des machines massivement parallèles pour l'usage général est encore très limité, par manque d'environnement de programmation. Les recherches en cours dans le domaine de la parallélisation et de la distribution automatiques de programmes visent à combler cette lacune.

En matière de systèmes d'exploitation, ces machines sont d'ores et déjà dotées de systèmes identiques à ceux que l'on trouve sur des réseaux de stations de travail (OSF/1, noyaux Mach, Chorus, ...), offrant ainsi à l'utilisateur une interface UNIX.

Lorsque la machine parallèle est utilisée, comme c'est le cas généralement, en tant que serveur spécialisé, le surcoût apporté par le système doit être minimum afin de délivrer le maximum de performance au niveau du calcul. La fonction "contrôle de l'exécution" est ici dominante, voire la seule présente. En effet, le développement des programmes peut être effectué sur des stations de travail indépendantes de la machine parallèle, excluant le besoin d'une fonction "environnement de programmation" sur celle-ci. Une autre alternative consiste à dédier certains processeurs de la machine parallèle aux fonctions traditionnelles du système d'exploitation : gestion de ressources, aide à la programmation.

Françoise André, professeur à l'Université de Rennes I, IRISA (URA 227 CNRS), campus de Beaulieu, 35042 Rennes Cedex.

LES RÉSEAUX INFORMATIQUES

Les ordinateurs, qu'ils soient grands ou petits, communiquent de plus en plus entre eux. Ils sont réunis dans des réseaux dont la technologie et les performances sont en constante évolution.

COMPUTER NETWORKS - By about the year 2000, cell-switching should end the battle between communication via circuits and packet-switching. But the the competition between public and private networks will continue. In any case, and although a number of theoretical problems (in particular the issue of acknowledgement) have yet to be solved, computer networks should exert a cohesive influence on computing in general.

Michel Diaz
Guy Pujolle

Un ordinateur ou même un micro-ordinateur isolé n'est plus concevable actuellement. Les ordinateurs sont de plus en plus souvent reliés entre eux dans un réseau : la fonction de communication devient primordiale en informatique.

Depuis une vingtaine d'années, deux grandes techniques se disputent la suprématie du transport des données :
- la commutation de circuits, qui met en place un circuit physique réel entre l'émetteur et le récepteur (comme pour le téléphone). Les éléments binaires sont émis sur ce circuit. L'inconvénient majeur de cette solution est le coût élevé à payer ;
- la commutation de paquets, qui rassemble les données en blocs d'informations, nommés paquets. Ces paquets sont envoyés vers des nœuds de commutation où ils sont stockés, puis envoyés vers un nœud suivant, jusqu'à atteindre le destinataire.

La différence entre ces deux procédures se comprend facilement par analogie avec la transmission des lettres. La transmission par circuit est analogue à l'usage du télécopieur qui occupe en permanence une ligne, alors que la transmission de paquets ressemble au courrier classique où le même facteur dessert plusieurs destinataires.

Le choix entre ces deux modes dépend des applications. Le circuit est adapté à la transmission continue comme la parole numérique. Le paquet convient au transfert de données informatiques qui arrivent par à-coups (une ligne, une page...).

Bientôt la communication par cellules

Les réseaux longue distance des années 1990-2000 font appel simultanément aux circuits et aux paquets. Cette solution bâtarde sera remplacée vers les années 2000 par une commutation unique, la commutation de cellules. La cellule est un très petit paquet, de 53 octets (1 octet = 8 bits). La commutation de cellules se rapproche aussi de la commutation de circuit : la cellule, étant très petite, peut être remplie et envoyée instantanément. On ne perd pratiquement aucun temps dans sa gestion : cette technique possède donc potentiellement les avantages des deux approches.

Différents types de réseaux

Suivant la distance entre les points les plus éloignés qui doivent être reliés, le réseau est construit de façon différente. Les réseaux locaux LAN (*Local Area Network*) desservent typiquement un immeuble ou un groupe d'immeubles rapprochés, les réseaux à grande distance WAN (*Wide Area Network*) l'ensemble d'un pays. Les réseaux MAN (*Metropolitan Area Network*) sont intermédiaires. Ils sont adaptés à la desserte d'une ville.

Les réseaux locaux sont de deux types principaux : ceux qui ne desservent que des ordinateurs personnels, et ceux qui peuvent faire intervenir un ordinateur plus puissant.

Ces réseaux locaux, dits souvent d'entreprise, se distinguent par le débit impor-

Une station de travail multimédia (photo D. Daurat, LAAS, Toulouse).

tant : 10 Mbit/s pour Ethernet, entre 4 et 16 Mbit/s pour Token Ring, jusqu'à 100 Mbit/s pour FDDI (*Fiber Distributed Data Interface*) qui utilise les fibres optiques.

Quelquefois, il suffit d'employer un autocommutateur privé utilisable aussi bien pour la voix que pour les données. Mais il est limité aux débits bas de 64 kbit/s.

Ces réseaux sont actuellement confrontés à des demandes importantes d'augmentation de débit avec l'arrivée des transmissions multimédias (signaux informatiques, images, parole).

Les réseaux métropolitains MAN permettent de relier des points distants de quelques dizaines de kilomètres. Deux normes sont en concurrence : FDDI déjà rencontrée au niveau local et DQDB (*Distributed Queue Dual Bus*).

Réseaux publics ou privés ?

Selon les pays, les réseaux à longue distance sont publics ou privés. Dans notre pays, France Telecom est l'opérateur unique. Cette société propose Transpac orienté vers le paquet, et la gamme Transcom, Transdyn et Transfix orientée vers le circuit. Numeris occupe une position intermédiaire.

Des réseaux privés utilisant les architectures des grands constructeurs informatiques (SNA pour IBM, DSA pour Bull, DNA pour Digital…) peuvent également être mis en place sur des longues distances. Une harmonisation vers la norme OSI est en cours, ainsi que la mise au point de passerelles permettant le passage des informations venant d'un type de réseau sur un autre.

Coopératif et multimédia

Le but à très long terme est d'obtenir un seul et même réseau capable d'accepter tous les types d'applications. Les deux mots-clés sont *coopératif et multimédia*. Pour y arriver, le chemin est encore long mais les recherches sur les premières techniques de base s'accélèrent. Il reste à proposer les protocoles logiciels de haut niveau, de montrer que tous les cas de fonctionnement sont pris en compte (panne, erreur, catastrophes diverses…) et de dimensionner l'ensemble pour que les performances puissent être satisfaites.

Le problème n'est pas simple puisqu'il s'agit de transmettre à très grande vitesse des éléments binaires groupés en ensembles ayant une dimension variable, de quelques octets à quelques kilo-octets.

Un des logiciels en développement est le protocole de transport XTP (*eXpress Transfert Protocol*) qui s'est fixé les quatre objectifs suivants :
- gérer l'ensemble des interconnexions entre tous les utilisateurs et les effets de toutes les erreurs possibles ;
- assurer les diffusions efficaces de n émetteurs vers m récepteurs ;
- lier la vitesse d'émission de l'émetteur aux possibilités de réception des récepteurs, en calculant un flux variable dynamique selon les possibilités des récepteurs et celles des ressources de transfert ;
- fournir à l'utilisateur la meilleure qualité de service en optimisant les caractéristiques des réseaux disponibles.

Dans le cas général, il faudra trier les paquets selon les utilisateurs des réseaux, et traiter conjointement les différents paquets liés à un même utilisateur. Ces traitements complexes nécessitent des logiciels qui font à l'heure actuelle l'objet d'un grand nombre d'études. Il s'avère très difficile de satisfaire simultanément des contraintes de performances et des fonctions sophistiquées.

Les études actuelles portent, d'une part, sur la conception de réseaux pouvant supporter des débits de l'ordre du gigabit/s et, d'autre part, sur la conception de protocoles permettant d'assurer le transfert isochrone d'objets multimédias.

En conséquence, l'architecture matérielle et logicielle des réseaux devrait constituer dans les années à venir un des systèmes les plus complexes construits par l'homme.

Michel Diaz, directeur de recherche au CNRS, Laboratoire d'automatique et d'analyse des systèmes (UPR 8001 CNRS), CNRS, 7, avenue du Colonel Roche, 31077 Toulouse.

Guy Pujolle, professeur à l'Université Pierre et Marie Curie, directeur de l'unité Méthodologie et architecture des systèmes informatiques (URA 818 CNRS), Université Pierre et Marie Curie, Tour 65-66 E2, 4, place Jussieu, 75252 Paris Cedex 05.

PROTOCOLE ET FIABILITÉ

Le problème dit *des deux armées* est célèbre dans le milieu des transmissions.
Deux armées A et B sont situées sur des positions telles que l'armée A est décomposée en deux parties A1 et A2, séparées par l'armée B ; de plus, le général en chef de A se trouve dans la partie A1. Les hypothèses sont les suivantes : si l'armée A au complet (c'est-à-dire A1 et A2 simultanément) attaque B, elle gagne le combat; si par contre A1 et A2 attaquent B en ordre dispersé, l'armée A perd le combat. Le problème consiste à concevoir des règles (appelées protocoles dans le contexte des réseaux, et algorithmes répartis dans un cadre plus général) qui permettent au général en chef de A d'attaquer B et de vaincre ; il a pour cela à sa disposition des messagers qu'il peut envoyer à A2 munis d'informations, mais ceux-ci peuvent être faits prisonniers et ne jamais parvenir à A2. Il est donc nécessaire que A2 renvoie un accusé de réception (à l'aide d'un messager qui va de A2 vers A1) afin d'indiquer la bonne réception des informations émises par A1 : à la réception d'un tel accusé, A1 sait que A2 a reçu ses directives mais A2 n'a pas les moyens de savoir que A1 est ou sera muni d'une telle connaissance : en effet si le messager transportant l'accusé est fait prisonnier, il ne parviendra jamais à A1 et A2 ne sait pas si le messager a été ou non fait prisonnier. Il faut donc envisager un accusé de réception de A1 vers A2 pour que A1 indique à A2 la bonne réception de l'accusé précédent. Il n'est pas difficile de voir qu'avec ce type de règles, il faudra une infinité de messagers (transportant des accusés d'accusés… de réception) avant que l'armée A puisse prendre la décision d'attaquer.
Aucune solution n'existe pour un problème posé en ces termes, qui apparaît pourtant chaque fois que deux interlocuteurs conversent par l'intermédiaire d'un canal non fiable.
Cet exemple simple illustre la nécessité d'étudier les algorithmes répartis tant sur le plan théorique que pratique. Le caractère asynchrone des canaux de communication qui véhiculent les messages, les vitesses des processeurs (les entités de calcul) et l'irruption de défaillances constituent les difficultés majeures auxquelles se trouvent confrontés les concepteurs d'algorithmes répartis. On a pu ainsi dire que si la finalité des algorithmes parallèles était la recherche de l'efficacité (souvent mesurée en temps de calcul), celle des algorithmes répartis est la maîtrise de l'incertitude sur l'état du calcul que créent l'asynchronisme des supports (canaux et processeurs) et l'occurence de défaillances.

Michel Raynal, professeur à l'Université de Rennes I, IRISA - INRIA, Campus de Beaulieu, 35042 Rennes Cedex.

SURETÉ DE FONCTIONNEMENT ET TOLÉRANCE AUX FAUTES

Le matériel informatique, comme les programmes, n'est pas sans défauts. Pour que le comportement final reste toujours bon, il est nécessaire d'utiliser des systèmes qui tolèrent des fautes.

Jean-Claude Laprie

La sûreté de fonctionnement, définie comme la propriété d'un système telle que ses utilisateurs puissent placer une confiance justifiée dans le service qu'il leur délivre, est un concept générique englobant les notions de fiabilité, disponibilité, sécurité-innocuité (au sens de l'évitement de défaillances catastrophiques), sécurité-confidentialité (au sens de la prévention d'accès ou de modifications non autorisés de l'information).

Tolérer les fautes

Tolérer les fautes nécessite en premier lieu d'éliminer les erreurs produites par l'activation de fautes internes (bogues de logiciel, collage d'une connexion matérielle à une valeur binaire...) ou l'occurrence de fautes externes (perturbation électro-magnétique, action inappropriée d'un opérateur...), si possible avant qu'une défaillance ne survienne. Une faute permanente nécessite de plus d'être diagnostiquée et inactivée afin d'éviter qu'elle ne se reproduise. La tolérance aux fautes est (comme tout concept lié à la sûreté de fonctionnement) un concept récursif : les mécanismes destinés à mettre en œuvre la tolérance aux fautes doivent être protégés contre les fautes susceptibles de les affecter eux-mêmes.

Ce qui précède s'applique tant aux fautes physiques (consécutives à des défaillances opérationnelles du matériel), qu'aux fautes de conception (résultant d'erreurs commises dans le développement des matériels et, surtout, des logiciels) : les classes de fautes qui peuvent être réellement tolérées par un système donné dépendent des hypothèses de fautes considérées dans le processus de conception, ce qui est conditionné par l'indépendance, par rapport aux processus de création et d'activation des fautes, des redondances nécessaires pour le traitement des erreurs et des fautes. Un exemple est donné lorsque l'on considère une méthode (largement utilisée), consistant à effectuer des traitements multiples par des voies de traitement elle-mêmes multiples. Lorsque la tolérance aux fautes physiques seules est recherchée, les voies multiples peuvent être identiques, en vertu de l'hypothèse selon laquelle des composants matériels défaillent indépendamment; une telle approche, de toute évidence, n'est pas adéquate pour la tolérance aux fautes de conception où les multiples voies doivent délivrer des services identiques via des conceptions et des réalisations séparées, approche que l'on désigne par la notion de diversification fonctionnelle ou dissymétrie.

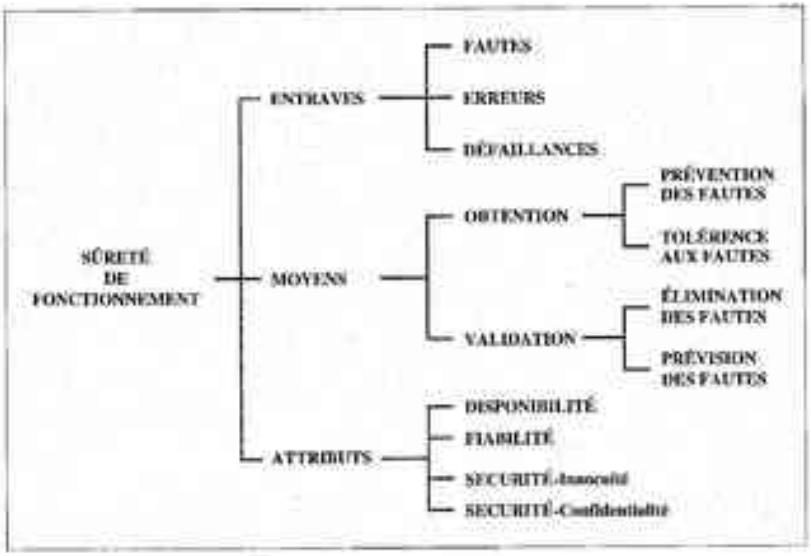
La tolérance aux fautes n'est pas limitée aux fautes accidentelles. Certains mécanismes de détection d'erreur sont destinés tant aux fautes accidentelles qu'aux fautes intentionnelles (par exemple, les techniques de protection d'accès aux mémoires), et des approches ont été proposées pour tolérer à la fois les intrusions et les fautes physiques (défaillances accidentelles), via la fragmentation des objets à protéger, la réplication des fragments et leur dissémination sur les machines d'un réseau.

Perception des classes de fautes par les utilisateurs

Les fautes physiques peuvent être considérées comme étant – relativement – maîtrisées à l'heure actuelle, où les sources de défaillance des systèmes informatiques les plus préoccupantes, tant en fréquence qu'en sévérité, sont les fautes de conception des logiciels et les fautes humaines d'interaction (consécutives à des erreurs d'opérateurs ou d'équipes de maintenance). Parallèlement à ces fautes accidentelles, les fautes intentionnelles ont vu leur importance croître, en particulier en raison du développement de l'informatique répartie. Tolérer simultanément les fautes accidentelles et les fautes intentionnelles constitue le défi actuel.

Jean-Claude Laprie, directeur de recherche au CNRS, directeur du Laboratoire d'ingénierie de la sûreté de fonctionnement, (Laboratoire commun LAAS - Matra Marconi Space - Technicatome), 7, avenue du Colonel Roche, 31077 Toulouse Cedex.

DEPENDABILITY AND FAULT-TOLERANCE
Dependability is a generic concept encompassing the notions of reliability, availability, safety and security. Fault tolerance is one of the means for providing dependable computing systems, where a very widespread method is to perform multiple, independent processing.



L'arbre de la sûreté de fonctionnement.

LA MODÉLISATION DES SYSTÈMES INFORMATIQUES

La modélisation et l'analyse quantitative des systèmes informatiques sont aujourd'hui des étapes essentielles de la conception des architectures et des logiciels. Ces techniques permettent de prévoir le comportement d'un système en cours de conception et de modifier sa structure pour qu'il vérifie certaines propriétés quantitatives ou qualitatives.

MODELING DP SYSTEMS - It has become essential to use modeling methods before designing hardware or software. Only by doing so can operational capabilities be met. Queuing networks and Petri networks are currently the most commonly used modeling formalisms.

François Baccelli

Chacun s'accorde pour situer les débuts de cette discipline dans le domaine des télécommunications, notamment dans les travaux d'A.K. Erlang (1917). Un des problèmes sur lesquels travaillait Erlang est celui de la probabilité de rejet dans un central téléphonique. Des communications arrivent avec une intensité λ, mais de manière irrégulière et imprévisible, à un central téléphonique comportant K canaux. Si un appel trouve tous les canaux occupés, il est rejeté. Dans le cas contraire, on lui alloue un canal qu'il conserve pendant toute la durée de la communication.

Les durées des communications sont elles aussi imprévisibles, on connaît cependant leur moyenne m. La non-prévisibilité des durées et des dates de communications, qui résulte du libre arbitre des abonnés, est une caractéristique essentielle du problème. Quelle est, dans ces conditions, la fréquence des communications rejetées ?

Pour résoudre ce problème, Erlang se place dans un cadre probabiliste, qui permet de prendre naturellement en compte et de quantifier le caractère imprévisible des dates de communications et de leurs durées. Sous des hypothèses mathématiques que nous ne préciserons pas en détail ici, Erlang montre que la probabilité de rejet d'une communication est donnée par la formule :

$$p = \frac{(\lambda m)^K/K!}{\sum_{i=0}^{K}(\lambda m)^i/i!},$$

qui porte son nom et qui est encore utilisée en téléphonie et plus généralement pour l'étude des systèmes ainsi formés d'une station avec rejet.

Beaucoup de chemin a été parcouru entre ces travaux d'Erlang et l'optimisation des calculateurs multi-programmés et notamment celle des procédures de temps partagé que nous utilisons aujourd'hui, obtenues vers le milieu des années 70. Néanmoins, cette étape décisive dans le développement de l'informatique était basée sur une analyse quantitative qu'on peut voir comme le prolongement direct de la démarche de pionnier d'Erlang.

Dans le modèle à serveur central (voir figure) que nous décrivons maintenant, un système informatique multiprogrammé est représenté comme un réseau de stations interconnectées, parcourues par des clients dont nous préciserons la nature dans un instant. Une station peut être une unité centrale, un disque, une unité d'entrée-sortie etc.

Les stations jouent un rôle similaire à celui des canaux dans l'exemple du modèle d'Erlang. Les clients de ce réseau de stations sont les programmes qui s'exécutent sur ce système. Ils sont à mettre en parallèle avec les communications du modèle d'Erlang. Chacun de ces programmes doit recevoir certains services de ces stations lors de son exécution. Chaque programme peut être vu comme une entité qui migre d'une station à une autre pour y recevoir les services dont elle a successivement besoin. Par exemple, un programme peut successivement avoir besoin d'un service de l'unité centrale (exécution d'un certain nombre de lignes de code sans accès mémoire), puis d'un service de tel disque (accès mémoire), puis de nouveau de l'unité centrale etc. Chaque station doit donc traiter un flux de demandes de services émanant des divers programmes en cours d'exécution. Si plusieurs programmes demandent la même station à un instant donné, un mécanisme de file d'attente est mis en œuvre : l'un d'entre eux reçoit le service qu'il demande, et les autres sont mis en attente. De même que les communications sont de durée imprévisible pour le concepteur du central téléphonique, le plus souvent, la seule connaissance a priori que l'on peut avoir sur le comportement des programmes est de nature statistique.

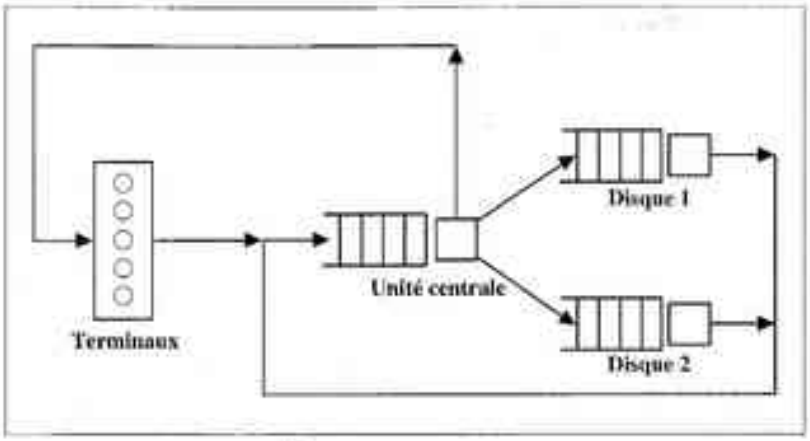
Les questions les plus importantes concernant ce modèle sont celles relatives au nombre maximal de programmes qu'il convient de laisser s'exécuter sur la machine (ou encore taux de multiprogrammation), tout en respectant des normes de fonctionnement acceptables pour l'utilisateur (par exemple un temps de réponse moyen de X secondes). Il se trouve que sous des hypothèses probabilistes adéquates, le modèle à serveur central se résout assez simplement, et qu'on peut notamment obtenir des formules du genre de celle d'Erlang pour calculer les temps de réponse moyens des programmes en fonction de leur nombre, des statistiques des durées d'exécution des programmes sur chacune des stations, et des fréquences de migration. On sait aussi déterminer simplement le nombre moyen des programmes en attente dans chacune des stations, ce qui permet de localiser les goulots d'étranglement du système (les ressources sous-dimensionnées qui ont un grand nombre de programmes en attente).

Quels sont les formalismes propres à la représentation de systèmes comme ceux que nous venons de décrire ?

Réseaux de files d'attente et réseaux de Petri

Le premier formalisme utilisé fut celui des réseaux de files d'attente. Ce formalisme connaît actuellement un succès industriel indéniable, comme en témoignent les nombreux produits logiciels fondés sur les réseaux de files d'attente actuellement disponibles sur le marché. S'il est relativement simple à manipuler, ce formalisme possède un certain nombre de défauts. En particulier, il doit être enrichi afin de représenter les divers mécanismes de synchronisation propres au contrôle d'exécutions parallèles dans les systèmes multiprocesseurs.

On peut voir le formalisme des réseaux de Petri comme une version plus abstraite et plus générale du précédent. Initialement utilisés pour la preuve de ▶



Le modèle à serveur central.

programmes, ces réseaux sont incontestablement un pas en avant par rapport au formalisme précédent. Leur puissance de description est à peu près équivalente à celle des machines de Turing, et quelques éléments de base combinés de manière adéquate permettent de construire la plupart des objets usuels manipulés par la théorie des réseaux de files d'attente, et bien d'autres encore. Ce formalisme souffre cependant d'une relativement faible diffusion industrielle et d'une difficulté de manipulation accrue par rapport au précédent.

Une fois le modèle défini, et ses paramètres fixés, il faut pouvoir le résoudre, c'est-à-dire calculer de manière efficace les caractéristiques du système telles que les temps moyens de réponse, les probabilités de rejet, les tailles moyennes des files d'attente, etc. Ceci n'est pas toujours facile, puisque ces systèmes conjuguent les trois difficultés intrinsèques suivantes : ce sont des systèmes dynamiques, leur dynamique est non standard et aléatoire.

Outre les méthodes de résolution fondées sur les solutions exactes, qui forment aujourd'hui une branche très vivante du calcul des probabilités, nous évoquerons celles fondées sur la simulation à événements discrets, qui consistent à calculer pas à pas l'évolution aléatoire du système étudié et à faire des statistiques sur son évolution pour obtenir des estimateurs des quantités que l'on cherche à déterminer.

Calcul des probabilités

L'un des fleurons de la théorie analytique des systèmes à événements discrets est la théorie des réseaux à forme produit, qui a des applications spectaculaires tant dans le domaine des réseaux de files d'attente que dans les réseaux de Petri stochastiques. C'est cette théorie qui sous-tend l'analyse du modèle à serveur central. C'est elle qui est utilisée de manière courante pour le dimensionnement des grands réseaux à commutation de paquets ou de circuits. C'est sur elle encore que sont fondés tous les solveurs analytiques des logiciels d'évaluation de performances actuellement sur la marché. Le succès de cette théorie vient essentiellement de sa simplicité. Pour les réseaux de files d'attente ou les réseaux de Petri satisfaisant les conditions du théorème de la forme produit, la distribution du nombre des objets dans chaque station est donnée comme le produit de distributions géométriques (éventuellement tronquées) dont les paramètres sont facilement calculables. Hors de cette classe, les résultats analytiques sont beaucoup moins nombreux, même s'il existe des théories assez complètes des problèmes à une et à deux files.

Une autre approche consiste à étudier les propriétés algébriques de ce type de dynamique. Dans ce cadre, les méthodes fondées sur la théorie ergodique et sur la théorie des processus ponctuels ont permis de dégager une approche générale pour le calcul des conditions de stabilité et des débits dans certains réseaux de Petri sans contrainte de dimension. On sait maintenant caractériser une classe de réseaux de Petri stochastiques pour lesquels la théorie de la stabilité se ramène à des calculs d'exposants de Lyapounov, semblables à ceux des systèmes classiques.

Les problèmes du contrôle de ces systèmes sont évidemment encore plus difficiles que ceux posés par l'analyse directe évoquée ci-dessus. Un exemple typique est le suivant, motivé par le problème de l'admission de nouvelles communications dans les réseaux numériques à intégration de service : considérons un central comme celui évoqué dans le cas du modèle d'Erlang, mais avec des communications de diverses classes. Certaines communications sont des appels téléphoniques codant de la voix à 64 kilobits par seconde. D'autres encore transportent des images, éventuellement animées, qui requièrent le transfert de plusieurs mégabits par seconde. Pour une communication de type voix, un seul canal suffit. Pour une communication du deuxième type, il faut N canaux simultanément. S'il reste N canaux libres et qu'une communication codant de l'image se présente, vaut-il mieux l'accepter au prix du rejet ultérieur d'un grand nombre de communications de voix, ou l'accepter ?

Simulation à événements discrets

Les problèmes scientifiques posés par cette approche sont d'abord ceux de la validité statistique des résultats. Typiquement, dans cette approche, on calculerait la probabilité de rejet du modèle d'Erlang en faisant évoluer pas à pas le modèle pendant une longue durée pendant laquelle on compterait les occurences de rejets. Un théorème de type ergodique permet en général de dire que cette mesure est un bon estimateur de la probabilité de rejet que l'on souhaite déterminer. Le problème difficile est le calcul de la vitesse de convergence de cet estimateur. Combien de temps doit durer la simulation pour que l'estimée du paramètre soit proche de la valeur réelle ?

La simulation, pour universelle qu'elle soit, ne répond cependant pas à tous les problèmes posés : en particulier, on ne sait pas estimer les probabilités des événements rares (tout simplement parce qu'ils sont d'occurence rare et qu'on ne les voit donc pas arriver dans les simulations). De même, les procédures d'optimisation paramétriques de type gradient, fondées sur la simulation, restent bien souvent impraticables, à cause de la lenteur des algorithmes de calcul des évolutions pas à pas.

Des directions nouvelles en simulation ont cependant émergé ces dernières années. La première concerne le calcul de gradients évoqué ci-dessus. La méthode de base consistant à faire tourner une première simulation pour la valeur λ du paramètre, puis une autre pour la valeur $\lambda + \delta\lambda$ n'est évidemment pas la meilleure, bien qu'on l'utilise couramment. On montre que dans les bons cas, le calcul d'une seule longue évolution du système, disons pour λ, est assez riche en information pour déterminer le comportement du système pour $\lambda + \delta\lambda$ aussi. Ces résultats devraient avoir un impact important sur les techniques d'optimisation par simulation.

Une autre direction nouvelle pour diminuer le coût en temps de la simulation

est sa parallélisation. Une floraison de nouveaux algorithmes de simulation parallèle a vu le jour dans les dix dernières années. Des accélérations de l'ordre de 10^3 peuvent être obtenues pour certaines classes de systèmes.

François Baccelli, directeur de recherche à l'INRIA, Sophia-Antipolis, 2004, route des Lucioles, 06565 Valbonne Cedex.

Le formalisme et les résultats de la théorie des réseaux de files d'attente se sont d'abord traduits, vers la fin des années 70, par l'apparition de logiciels d'évaluation généralistes, comme RESQ chez IBM et QNAP2 développé par l'INRIA et Bull. Ultérieurement sont apparus des logiciels spécialisés, comme PANACEA, QNA et Q+, tous développés chez AT&T. Dans le domaine des réseaux de Petri, les logiciels sont plus récents et sensiblement moins diffusés. Citons toutefois GREAT-SPN actuellement développé par l'Université de Turin et DESIGN-CPN par Meta Software Corporation. Plus récemment, un logiciel européen d'évaluation de performances appelé IMSE (*Integrated Modelling Support Environment*) a été réalisé dans le cadre d'un projet ESPRIT2. Ce logiciel intègre divers formalismes dont les réseaux de files d'attente et les réseaux de Petri, de nombreux outils d'analyse tels que la simulation et les méthodes analytiques, le tout sous un environnement graphique commun.

En France, la recherche publique est concentrée au Centre National d'Etudes des Télécommunications (analyse et contrôle des réseaux ATM), à l'INRIA (simulation, méthodes analytiques et contrôle), au CNRS (contrôle des réseaux) et dans de nombreuses universités. La recherche dans le secteur privé est à la fois présente dans les grands groupes (notamment Thomson, Bull, IBM, Sagem) et dans des startups spécialisées dans ces techniques comme Simulog ou Verilog. Des projets de recherche coordonnée sont financés par le MRE dans le cadre du PRC C3 qui rassemble des équipes universitaires d'Orsay, de Grenoble et de Paris entre autres, ainsi que dans le cadre d'une *Basic Research Action* nommée QMIPS (*Quantitative Modeling in Parallel Systems*) qui rassemble des équipes universitaires et des centres de recherche de six pays européens.

LES SYSTÈMES DE BASES DE DONNÉES

Vers 1960, les systèmes de gestion de fichiers ont laissé la place aux systèmes de bases de données. Trois modèles se sont succédé : les modèles réseaux ou hiérarchiques qui sont sur le déclin, le modèle relationnel qui est à son maximum et commence à être concurrencé par le modèle objet.

Claude Delobel

DATA BASE SYSTEMS - The need to process new types of data together with the increased size of data have given rise to object-oriented data base technology. The relational model which has now superceded the network and hierarchical models of the 60's will therefore also become obsolete in the long term despite its benefits, in particular with the generalization of the SQL query language.

La fonction d'un système de base de données est d'offrir à un groupe d'usagers la possibilité de traiter à l'aide de programmes informatiques une grande masse d'information structurée. Il garantit aussi la cohérence, l'intégrité, la confidentialité et la sécurité des données.

L'évolution, du système de gestion de fichiers au système de base de données, a été possible parce que la technologie a offert vers 1960 les premiers éléments permettant de construire de tels logiciels. Les données pouvaient être stockées sur des unités de disques où l'accès est plus rapide que sur des bandes magnétiques. Les systèmes d'exploitation à temps partagé faisaient leur apparition, permettant ainsi de traiter plusieurs activités qui apparaissent simultanées pour un observateur extérieur. Enfin, l'accès à l'information pouvait se faire en utilisant des terminaux connectés à l'ordinateur central. Aujourd'hui, les unités de disque permettent de stocker de l'ordre du gigaoctet. La palette des supports physiques s'est élargie : disquettes, disques optiques, enregistrements vidéo. Au concept d'ordinateur central se substituent de plus en plus des unités multi-processeurs utilisant des systèmes d'exploitation répartie, accroissant de manière significative l'exécution parallèle des traitements et la sécurité globale de fonctionnement en cas de panne d'une unité physique. Enfin, le développement des techniques de transmission des données à haut débit et l'apparition des stations de travail ont modifié l'architecture des matériels informatiques.

Les systèmes de bases de données se sont non seulement adaptés à l'évolution technologique des architectures des systèmes et des matériels, mais ils ont subi des évolutions profondes dans leurs architectures internes. On s'accorde à considérer qu'il y a eu trois générations (Fig. 1) successives de systèmes de bases de données : un modèle réseau ou hiérarchique, un modèle relationnel et un modèle objet.

La génération des années 60

La première génération, née vers 1962 avec les modèles réseaux ou hiérarchiques (voir l'article de S. Krakowiak), permet à l'utilisateur de structurer ses données dans des fichiers reliés entre eux par des liens de dépendance, de type fichier maître à fichier membre (ou relation parent-enfant). Dans un modèle hiérarchique, ce principe est répété pour ne construire que des structures arborescentes, alors que le modèle réseau offre une structure comparable à celle d'un réseau. Le programmeur "navigue" dans la base de données. Il y a donc une grande dépendance entre l'organisation logique des traitements et l'organisation physique des données. Toute modification dans l'organisation des données entraîne une modification plus ou moins lourde de l'écriture des programmes qui est, de plus, difficile : pour exprimer un besoin simple tel que "combien y a-t-il d'employés âgés de plus de 57 ans dans le département des jouets ?", il faut faire appel à un spécialiste.

Celle des années 70

La deuxième génération, née au début des années 70 des travaux de E. F. Codd, a été appelée modèle relationnel. Les données de l'utilisateur sont représentées dans des tables de valeurs (des relations, d'où

le nom) sans aucun lien physique entre elles. Les questions sont posées dans un langage déclaratif, le plus souvent SQL (voir encadré) qui s'est imposé comme standard. Le programmeur d'application n'indique pas les accès à la base, il se concentre sur la logique de ses besoins et non pas sur la façon de l'obtenir. L'utilisation de ce langage conduit à une augmentation de la productivité car il est plus simple à écrire, plus simple à lire et plus facile à maintenir. En revanche, le système doit posséder un outil qui traduit ce langage en une séquence d'accès aux données et qui recherche, parmi toutes les traductions possibles, celles qui conduisent aux meilleures performances. Cet outil est un optimiseur d'accès.

Le modèle objet

La dernière génération, orientée objet, n'est pas encore complètement figée. Elle est adaptée aux problèmes nouveaux de grande complexité et de grande taille que l'on rencontre de plus en plus. Par exemple, la conception assistée par ordinateur d'un circuit VLSI nécessite d'archiver les étapes du processus de conception ainsi que les représentations des objets élaborés. La représentation du même objet est différente selon qu'il s'agit du schéma logique du circuit ou de ses caractéristiques électriques. De même, une base de données de produits chimiques décrit la structure des produits avec leurs propriétés, mais elle doit aussi contenir des règles permettant d'assembler de nouvelles molécules. Le séquençage du génome humain conduit à envisager des bases de données gigantesques (de l'ordre du téraoctet). Enfin, la gestion de la documentation technique nécessite d'archiver des données de nature différente : textes, dessins, images, sons.

Un changement qualitatif

Ces applications nouvelles ont deux caractéristiques : elles remettent en cause à la fois les mécanismes d'accès et la gestion des types de données. Les mécanismes qui étaient valables pour des fichiers de millions d'octets (mégaoctet) ne le sont plus quand ils en feront mille milliards (téraoctet), et chaque application a des besoins spécifiques. Par exemple, les techniques d'indexation utilisées dans une application géographique sont différentes de celles d'une application textuelle. Avec le modèle relationnel, les types de données proposés au programmeur sont limités, car seul le concept de table de valeurs existe. Pour satisfaire les nouveaux besoins, la gestion des types de données doit être plus souple et adaptée à chaque application. Les langages à objet

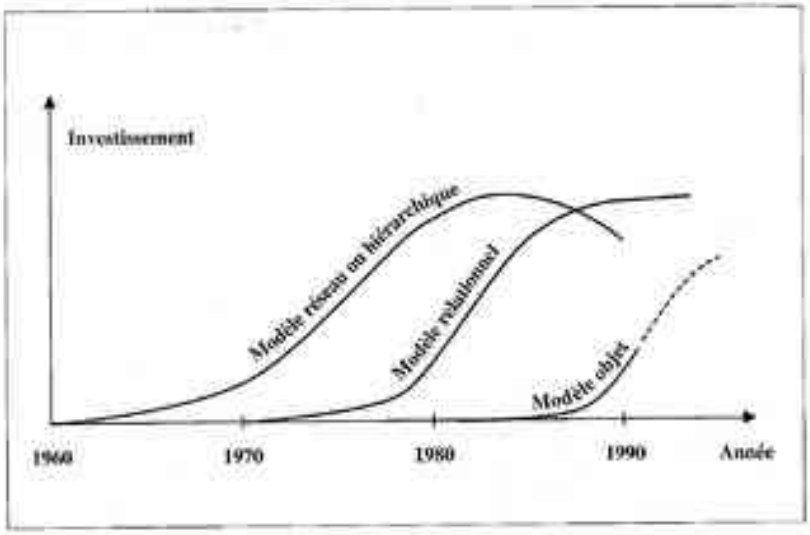


Fig. 1 - Les trois générations des systèmes de bases de données. Chacune correspond à un cycle de développement du logiciel. Il passe par quatre phases successives : recherche fondamentale, prototypage, industrialisation, et déclin. A partir des exemples vécus avec les deux premières générations, les deux premières phases ont eu une durée de cinq ans chacune, soit une période de dix ans avant d'atteindre la commercialisation qui comporte une phase de croissance avec la réaction progressive du marché, puis une phase de développement intensif en cas de succès. Avec la dernière génération, on constate un raccourcissement des deux premières phases puisque la phase de commercialisation a été obtenue cinq ans après le début de la recherche. Ce phénomène s'explique probablement par une meilleure maîtrise du développement des gros logiciels.

permettent de définir un système de typage spécifique à chaque classe d'applications, ce qui explique les efforts faits depuis quelques années pour utiliser ces langages pour construire et manipuler les bases de données, efforts qui dépassent le cadre de la recherche et trouvent des échos très favorables dans les milieux industriels les plus dynamiques.

Cependant, malgré cette orientation très forte, on ne peut négliger les apports de ce qu'on appelle les systèmes déductifs, dans lesquels les besoins des utilisateurs sont exprimés sous forme de règles, traduisant certaines propriétés invariantes ou certaines conditions susceptibles de déclencher telle action.

Lorsque le modèle relationnel a été proposé, il est apparu comme un cadre théorique élégant mais dont l'implémentation s'avérait totalement inefficace. La recherche a su inventer des langages d'in-

LE LANGAGE SQL

Le langage SQL permet de rechercher dans la base de données (Fig. 2) les éléments qui répondent à une question. Par exemple, "quelles sont les pièces composées qui utilisent des composants de type écrou ?". Pour répondre, on peut d'abord rechercher dans la table PIECE-BASE toutes les lignes dans lesquelles le mot "écrou" apparaît, et sélectionner le numéro de pièce de base correspondante. On obtient ainsi un ensemble de numéros. Dans une deuxième étape, on peut rechercher dans la table PIECE-COMPOSEE toutes les lignes où la pièce de base figure dans cet ensemble. Ces lignes répondent à la question. Le langage SQL va exprimer en une seule phrase les deux étapes de ce processus :

Select NPC
from PIECE-COMPOSEE
where NPB in select NPB
from PIECE-BASE
where TYPE = "écrou"

Si l'écriture de la phrase SQL suit le processus mental d'un utilisateur pour résoudre le problème, le système va suivre un autre chemin. Il recherchera parmi toutes les stratégies d'accès possibles, compatibles avec la logique contenue dans la requête, celle qui lui semble optimale. C'est dans la compréhension de ce problème que la recherche a permis de déboucher sur les techniques d'optimisation des langages de requêtes.

PIÈCE-COMPOSÉE

NPC	NPB	QTE
12315	1001	3
12315	1002	3
----	----	----
12355	1003	1

PIÈCE-BASE

NPB	PRIX	TYPE
1001	0.50	écrou
1002	0.52	boulon
----	----	----
1003	0.47	écrou

Les tables de valeur PIÈCE-COMPOSÉE et PIÈCE-BASE

Fig. 2 - Le modèle relationnel de la deuxième génération. Les données représentées dans les deux tables de valeur PIÈCE-COMPOSÉE et PIÈCE-BASE décrivent la composition d'un produit. La table PIÈCE-COMPOSÉE est construite sur trois colonnes : le numéro de pièce composée (NPC), le numéro de pièce de base (NPB) et la quantité (QTE). La première ligne exprime le fait que la pièce de numéro 12315 est composée de la pièce de base 1001 en quantité de 3. Tous les faits similaires sont regroupés dans la même table, et à chaque fait correspond une ligne. La deuxième table décrit la nature de chaque pièce de base en utilisant trois colonnes libellées NPB, PRIX et TYPE. Chaque ligne de la table a la même signification, ainsi la première ligne peut s'interpréter comme la "pièce de base 1001 est de type écrou et coûte 0,50 francs". La représentation sous forme de table est simple et correspond à un modèle visuel maîtrisé par beaucoup de personnes. De plus, il n'y a aucune caractéristique technique sur l'organisation des données. Le contenu de la base de données correspond au contenu des tables, tandis que le schéma de la base de données correspond aux descriptions des entêtes des tables ici pièce-composée et pièce-basel.

DE SOCRATE À ALTAÏR

Les relations entre la recherche et l'industrie du logiciel des bases de données sont un cas probablement à part. D'une part, il existe un marché parfaitement identifié et quantifié : la valeur annuelle du chiffre d'affaires a été évaluée pour 1990 à 3,9 milliards de dollars. D'autre part, les résultats de la recherche ont souvent été transférés rapidement vers l'industrie. La recherche en France a toujours joué un rôle d'avant-garde. Dès 1968, l'université de Grenoble lançait le projet Socrate qui allait devenir un des produits de la première génération. Une action similaire, et de plus large envergure, s'est renouvelée avec le GIP Altaïr vingt ans après. Le système O_2 (voir encadré), développé par Altaïr et commercialisé par la jeune société O_2Technology, est un des premiers systèmes orientés objet. De même, l'Université de Technologie de Compiègne a favorisé le développement de la société Graphaël qui distribue le système G-Base.

terrogation de haut niveau, développer la théorie et les algorithmes nécessaires à l'optimisation de ces langages, construire des protocoles de gestion des transactions et de reprise, et mettre en œuvre des techniques d'indexation pour accéder efficacement aux données. Aujourd'hui toute l'industrie du logiciel des bases de données utilise ces résultats.

On pourrait penser que la technologie de ces bases est suffisamment évoluée pour recentrer les actions de recherche sur des domaines nouveaux, car le développement naturel des matériels et des systèmes suffira à assurer les besoins des bases de données au-delà de l'an 2000. Cela serait probablement une grave erreur car la taille des problèmes à résoudre sera encore en avance d'un ordre de grandeur sur celle du matériel. Seule la mise en œuvre d'architectures nouvelles associées à des algorithmes nouveaux pourra faire face aux "téra-bases" dont l'industrie et les administrations auront besoin au début du siècle prochain.

Claude Delobel, professeur à l'Université de Paris-Sud, Laboratoire de recherche en informatique (URA 410 CNRS), Université de Paris-Sud, Bât. 490, 91405 Orsay Cedex.

LE SYSTEME O_2

Le système O_2 est un système de bases de données orienté-objet muni d'un environnement complet de développement d'applications. Cet environnement (O_2Tools) intègre un langage de quatrième génération orienté-objet (O_2C), un langage de requête orienté-objet (O_2SQL), et un générateur automatique d'interfaces utilisateur (O_2Look).

En O_2, tout repose sur l'unique notion d'objet. Un objet est une entité qui possède une valeur, qui peut être complexe et multimédia (textes, images, graphiques, sons, programmes, etc.), et que l'on manipule au travers d'un ensemble d'opérations appelées méthodes. Les objets ayant même type de valeur et même ensemble de méthodes peuvent être regroupés au sein d'une même classe.

O_2C est une extension du langage C dans laquelle on trouve la possibilité de définir des variables O_2, précédées du mot-clé O_2, d'accéder aux champs de ces variables ou de leur appliquer des méthodes, par l'opérateur ->, et d'itérer sur des collections, par la boucle *for/in*.

La persistance est gérée en O_2 par la déclaration de racines de persistance. Tout objet relié à une racine de persistance obtient automatiquement tous les bénéfices offerts par un système de bases de données : stockage, accès concurrents par plusieurs utilisateurs, sécurité, distribution, reprise sur panne, etc.

La méthode *edit* est une méthode prédéfinie de O_2Look qui génère un masque d'affichage pour n'importe quel objet de la base, quelles que soient sa taille et sa complexité. Lorsqu'un objet est affiché par la méthode *edit*, il est possible d'éditer chacun de ses champs, soit par manipulation directe, soit par un couper/copier/coller, et déclencher interactivement n'importe laquelle de ses méthodes.

O_2Tools, l'environnement de développement de O_2, intègre un navigateur/éditeur graphique dans le schéma et dans la base, un gestionnaire de sources, un compilateur incrémental, un chargeur dynamique, et un débogueur symbolique. L'intégration de tous ces outils permet le développement complet d'une application sans sortir de O_2Tools.

Bruno Poyet, ingénieur consultant à O_2Technology, 7, rue du Parc de Clagny, 78000 Versailles.

Programmer aujourd'hui et demain

Lors de l'apparition des premiers ordinateurs, l'informatique se résumait à la programmation et celle-ci était extrêmement rudimentaire. Des programmeurs, au service d'une machine très onéreuse aux performances très faibles et avec des interfaces pauvres (ruban ou cartes perforées), rédigeaient et mettaient au point des programmes en langage machine dans des conditions difficiles.

Cinquante ans plus tard, l'informatique a beaucoup changé. La plupart de ses utilisateurs ne voient plus l'ordinateur qu'à travers des interfaces conviviales qui permettent d'utiliser les capacités de la machine en ignorant à peu près tout de sa structure et de son fonctionnement internes, de la même façon qu'on peut conduire une voiture sans jamais ouvrir son capot ou bénéficier du chauffage central sans connaître la thermodynamique.

Cette évolution a été permise par la conjonction de deux phénomènes distincts mais qui interagissent fortement : le progrès de la technologie des circuits et la constitution d'une science informatique en progrès rapide. Si l'utilisation des moyens informatiques est aujourd'hui à la portée de tous, c'est que la connaissance des machines que devaient déployer les programmeurs des temps héroïques est incorporée aux logiciels dont sont équipées les machines modernes. Mais par ailleurs, ces logiciels extrêmement complexes, issus de la recherche, n'auraient pu être développés, et ne pourraient être utilisés, sans l'accroissement des capacités des machines en vitesse et en mémoire, permis par la technologie des circuits.

La programmation n'a pas disparu. Elle reste au contraire au cœur de l'activité informatique, mais elle a énormément évolué. Affranchie de la soumission à la structure des machines et isolée d'elle par des couches logicielles de plus en plus riches, elle est devenue plus abstraite et s'est tournée vers des applications de plus en plus complexes qui sont partie prenante de tous les secteurs de l'activité industrielle et socio-économique. L'exigence de qualité des logiciels augmente tout en étant toujours plus difficile à satisfaire du fait de leur complexité.

Un logiciel de grande taille ne peut être développé, utilisé de

Station de travail avec environnement multi-fenêtre, outil de travail actuel du programmeur (cliché A. Eidelman, INRIA).

façon fiable et éventuellement évoluer que s'il est suffisamment structuré. La structure d'un logiciel de grande taille n'est maîtrisable qu'à travers des outils logiciels appelés "environnements de programmation" qui permettent de développer les logiciels par l'assemblage de constituants souvent hétérogènes, dont il importe d'assurer la cohérence. Par ailleurs, la structure d'un logiciel doit évidemment s'appuyer sur celle de l'application, ce qui suppose que celle-ci soit correctement spécifiée et qu'il soit possible de vérifier l'adéquation du logiciel à cette spécification. C'est là un des enjeux importants de la recherche en informatique. La solution de ce problème est d'ailleurs très difficile en pratique. Enfin, la spécification formelle d'une application de grande taille étant elle-même de grande taille, elle serait inutilisable sans outils informatiques pour la manipuler. Les environnements de programmation du futur devront donc incorporer des outils pour manipuler des spécifications et produire des programmes en accord avec ces spécifications.

Dans la maîtrise de la complexité, les langages de programmation, qui sont les outils de base du programmeur, jouent évidemment un rôle fondamental. Depuis les débuts de l'informatique, ils ont énormément évolué pour s'adapter aux exigences croissantes de la programmation. Ils ont gagné en puissance d'expression et en clarté et ont peu à peu incorporé de nombreux concepts issus de la recherche. Ils se sont aussi diversifiés pour répondre à la variété croissante des applications et à certaines innovations technologiques comme le parallélisme. L'évolution des langages offre un reflet assez fidèle des progrès de l'informatique, dans la mesure où les concepts nouveaux qui sont introduits pour faciliter la structuration des programmes doivent, pour être utilisés réellement, être incorporés aux outils de base que constituent les langages, que ce soit en enrichissant les langages anciens ou bien en en créant de radicalement nouveaux.

LE CODAGE BINAIRE

Le système de numération binaire ne comporte que deux symboles 0 et 1 pour représenter les nombres entiers.

0 1 2 3 4 5 6 7 8 ...

0 1 10 11 100 101 110 111 1000 ...

Son principal désavantage est la longueur des notations. Ainsi 32768 s'écrit 1000000000000000. Pour représenter les caractères, on a choisi le code ASCII sur sept chiffres (128 caractères). La lettre A de code 65 est représentée par 1000001. De la même façon, les instructions (opérations élémentaires des ordinateurs) sont codées en binaire. Sur un Motorola 68000 le code binaire :

0000 0110 0100 0001 0000 0000 0000 0101

correspond à l'addition de la constante 5 au contenu du registre D1.

Machine IBM 7040 à transistors (archives IBM). Produite à partir de 1963, cette machine de 30 tonnes possédait une mémoire à tores de ferrite de 200 Koctets. Les entrées de données se faisaient sur cartes perforées et les sorties sur imprimante. Le cycle de base était de 2 microsecondes.

Guy Cousineau, professeur à l'École normale supérieure.

Patrick Sallé, professeur à l'Institut national polytechnique de Toulouse.

DE L'HEXADÉCIMAL AUX OBJETS

Les langages de programmation ont évolué en même temps que le matériel. Ils deviennent de plus en plus abstraits et indépendants de la structure de la machine.

Patrick Sallé

FROM HEXADECIMAL NOTATION TO OBJECTS - Programming has become more and more removed from machine language. Assembly language was superceded by FORTRAN and COBOL, which were then replaced by structured languages. Modular programming has drawn benefits from the concepts of genericity and inheritance. Future languages will increasingly make use of mathematical logic.

L'ordinateur est une machine prodigieusement rapide, mais sans initiative. Nu, il ne sait réaliser que quelques opérations arithmétiques, et encore faut-il lui indiquer sur quelles données et dans quel ordre. Pour compléter ce tableau peu encourageant, il faut ajouter qu'il ne sait traiter que des données binaires et que le programme doit, lui aussi, être rédigé en binaire.

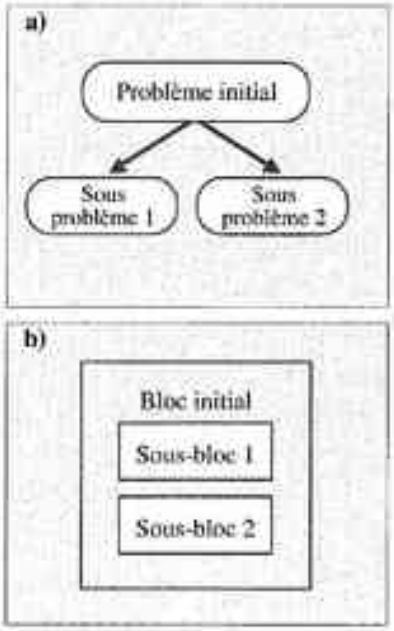
La solution trouvée par les informaticiens consiste à empiler des codages à partir du binaire jusqu'à arriver à des langages très puissants, très éloignés de la structure de la machine. Cet empilement de codages se retrouve dans la nature. Si l'on considère les particules élémentaires (électrons, protons, neutrons) comme un alphabet, leur assemblage permet la création d'un code plus riche, une centaine d'atomes. Sur ce nouvel alphabet, les molécules sont des mots et parmi celles-ci on trouve les quatre protéines qui sont l'alphabet du génome humain, lui-même à l'origine de notre alphabet... De même qu'il ne viendrait à personne l'idée de penser aux électrons qui sont à l'origine d'un poème, la plupart des informaticiens ne "descendent" plus jusqu'au binaire.

De l'assembleur à FORTRAN

La première étape a été celle des langages d'assemblage. C'est un codage quasi bijectif qui, à chaque instruction binaire, fait correspondre un code spécifique de chaque machine. Il permet également d'associer des noms symboliques aux emplacements mémoire, première étape vers la notion de variable. L'assembleur assure la traduction du langage d'assemblage en binaire. Les programmes en langage d'assemblage sont peu lisibles et difficiles à mettre au point. D'où la nécessité pratique d'utiliser un langage plus évolué.

Les premiers ordinateurs étaient rares, onéreux, lents et possédaient des mémoires très petites. Le premier souci du programmeur était la performance, c'est-à-dire utiliser le moins de mémoire possible et gagner des microsecondes. FORTRAN, langage scientifique et COBOL, langage de l'informatique de gestion, répondaient à ces critères. On constate une perte moyenne de 1,3 pour les performances d'un programme écrit en FORTRAN par rapport à celles d'un programme équivalent codé directement en langage d'assemblage. Un des intérêts de ces langages fut l'introduction de la notion de type. Chaque variable du programme désigne une donnée typée (entier, réel, caractère, ...) et dès lors il n'est plus possible d'écrire une expression qui additionne par erreur un caractère à une matrice par exemple.

FORTRAN est devenu rapidement une sorte de langage universel de tout ce qui n'était pas de la gestion. La taille des machines augmentant avec leurs performances, les programmes sont devenus de plus en plus gros et complexes. Des codes de 10 000, 100 000 puis un million de lignes sont apparus et les problèmes ont changé de nature. Tout d'abord les langages ne sont pas restés un outil de dialogue entre l'homme et la machine, mais sont devenus aussi un moyen de communication entre les programmeurs travaillant sur un même projet. Le besoin de performance à tout prix a diminué et c'est le temps humain passé pour la mise au point des programmes qui s'est mis à constituer la plus grande partie du coût de l'informatique. Pour des raisons économiques, il fallait rationaliser les méthodes de production de logiciels. Le principal défaut de FORTRAN est de favoriser l'écriture de programmes monolithiques avec une structure interne complexe. On a qualifié de programmation "spaghetti" ce style de programmation, pour exprimer l'impossibilité d'isoler un morceau de programme du fait de l'imbrication étroite de toutes ses parties. Or chacun sait qu'une bonne méthode pour résoudre un problème complexe est de le décomposer en sous-problèmes. D'où l'arrivée des langages de programmation structurée, dont PASCAL est le représentant le plus connu. Conçu initialement par Niklaus Wirth à des fins d'enseignement à l'Ecole polytechnique fédérale de Zürich, c'est un langage à structure de blocs. En programmation structurée, un problème correspond à un bloc du programme, et sa décomposition en sous-problèmes à l'emboîtement des blocs (voir figure).



Décomposition du problème a), et structure de blocs correspondante b).

Le programme gagne en lisibilité, car il peut être lu à plusieurs niveaux de détails. La décomposition logique du problème permet une distribution rationnelle du travail entre les membres d'une équipe de programmeurs.

Programmation modulaire

Avec PASCAL, la notion de type s'est enrichie. A partir des types de base, l'utilisateur peut créer ses propres types, par exemple le type nombre complexe ou date. Aux types de base sont associées des opérations prédéfinies, comme l'addition pour les entiers; de manière analogue, le programmeur peut définir des opérations sur ses propres types, l'addition de deux complexes, l'opération jour-suivant sur les dates... L'idée vient alors de structurer les programmes non seulement de manière algorithmique, mais également en termes de données, c'est-à-dire de regrouper ensemble toutes les parties d'un programme qui concernent le traitement

d'un type de données particulier. Il s'agit là d'un nouveau découpage logique des programmes, compatible avec celui de la programmation structurée. C'est l'objectif de la programmation modulaire. MODULA-2, successeur de PASCAL, et ADA sont les représentants les plus caractéristiques de cette programmation. L'idée sous-jacente est de réaliser des composants logiciels réutilisables, comme il existe des composants matériels. De même que l'on peut composer une chaîne hi-fi à l'aide de modules (amplificateur, lecteur laser, enceintes acoustiques ...), on peut bâtir un programme à l'aide de modules prédéfinis. L'analogie ne s'arrête pas là. Un amplificateur est une boîte noire qui possède des boutons accessibles de l'extérieur et une logique interne inaccessible. Un module logiciel, un gestionnaire de fichier par exemple, offre un certain nombre d'opérations accessibles (ouverture, lecture de fichier) et cache à l'utilisateur le détail de sa réalisation (comment les fichiers sont rangés sur le disque). Les avantages de cette structuration sont doubles : la réutilisation possible de modules standard et une mise au point séparée d'entités logiques autonomes.

Généricité et héritage

Cette programmation modulaire manque toutefois de souplesse. Il est rare qu'un composant préexistant convienne exactement. Deux solutions sont possibles et complémentaires pour rendre les modules plus adaptables aux besoins : la généricité et l'héritage.

La généricité permet de construire des modules paramétrables. De nombreux algorithmes peuvent être paramétrés, comme par exemple le tri d'un ensemble d'objets munis d'une relation d'ordre. On écrit un algorithme de tri tout à fait général dit générique. Au moment de son utilisation, il suffira de préciser le type des objets à trier et la relation d'ordre choisie pour engendrer automatiquement un programme exécutable. ADA est le premier langage commercialisé à avoir offert cette possibilité. Il est à noter que le traducteur, appelé compilateur du langage, intervient deux fois sur ces programmes. Le premier contrôle garantit que le programme engendré à partir de la version générique pourra être traduit sans erreur, à condition de fournir des paramètres bien typés. La deuxième intervention consiste à traduire une version particulière avec des paramètres donnés.

L'héritage autorise la construction de modules extensibles permettant à l'utilisateur d'ajouter des fonctionnalités supplémentaires à un code déjà existant. Il est lié à la notion d'objet introduite pour la première fois en 1967 dans le langage de simulation SIMULA.

Schématiquement, la définition d'un objet, appelée classe de l'objet, décrit sa structure (décomposition en sous-objets) et ses méthodes (les opérations qu'il peut effectuer sur lui-même). L'héritage permet de dériver une classe d'objets à partir de classes déjà existantes, de telle sorte que les objets de cette nouvelle classe héritent de toutes les propriétés des objets des classes héritées et possèdent des propriétés supplémentaires. Ainsi la structure des objets est enrichie par étapes et de nouvelles méthodes sont ajoutées.

Impératif, fonctionnel, déclaratif

Tous les langages ont évolué vers une structuration permettant de gérer une complexité croissante des programmes. On distingue toutefois trois manières différentes d'exprimer un algorithme donné selon que l'on fonde cette expression sur la logique mathématique, le calcul fonctionnel ou le schéma d'exécution de la machine.

Dans un langage impératif (FORTRAN, PASCAL, ADA, C,...), le programme décrit un enchaînement d'opérations élémentaires qui modifient les valeurs des variables du programme. Les langages impératifs sont les plus utilisés. Ils sont les plus efficaces, les plus faciles à traduire car relativement proches de la machine, mais ont une puissance d'expression assez faible. Les langages fonctionnels (LISP, ML) sont beaucoup plus concis mais moins rapides. Leurs domaines d'utilisation sont principalement la conception de prototypes et l'intelligence artificielle. Les langages déclaratifs (PROLOG) sont les plus abstraits et les plus expressifs. Ils sont surtout utilisés en intelligence artificielle. On y exprime les propriétés du problème à résoudre sous une forme très proche de la logique mathématique.

Si la notion d'algorithme semble stable, l'expression des algorithmes peut prendre des formes très diverses. L'évolution des performances et de la structure des machines peut rendre utilisables des techniques qui semblaient utopiques auparavant. Même s'il est périlleux de s'engager sur le devenir des langages à moyen terme, on constate que deux tendances se dégagent et font l'objet de travaux de recherche importants.

L'expérience a montré que nous ne savons pas développer des systèmes complexes sans que des erreurs de programmation ne se manifestent tôt ou tard par des pannes. On peut y remédier en construisant des systèmes à haut degré de fiabilité, capables de détecter l'occurrence des pannes et de restaurer automatiquement le système dans un état cohérent, dans l'attente d'une intervention humaine. Mais pour limiter au maximum les erreurs, les langages de programmation devront encore évoluer vers plus d'abstraction, de sorte que le programmeur puisse décrire ce que doit faire le système et non comment il le fait. Les futurs langages feront appel de plus en plus à la logique mathématique pour exprimer et prouver des propriétés.

Par ailleurs, nous savons construire des machines comportant un nombre important de processeurs de calcul qui travaillent en concurrence. Chacun de ces processeurs possède une architecture traditionnelle et est donc capable d'exécuter des algorithmes séquentiels. Mais les langages séquentiels sont mal adaptés à la programmation de ces machines multiprocesseurs qui peuvent faire simultanément plusieurs opérations. De nouveaux langages permettant d'exploiter le parallélisme du calcul sont donc à mettre au point. Une des voies consiste à répartir automatiquement les algorithmes séquentiels entre les processeurs.

Tous les langages de programmation sont "équivalents", et cela fut démontré mathématiquement bien avant l'invention des ordinateurs par le logicien Allan Turing, créateur d'une machine universelle abstraite. Plus précisément, si un langage est conçu avec un minimum de précautions, tout programme écrit dans un autre langage peut être traduit dans celui-ci et réciproquement.

CALCULABILITÉ

Est calculable tout problème qui peut être résolu en un nombre fini d'étapes dans un système formel. Par exemple en arithmétique, on peut calculer si un nombre n donné est la somme de deux nombres premiers. Lorsque le nombre d'étapes ne peut être borné, on parle de semi-calculabilité. Savoir si un nombre donné n est la différence de deux nombres premiers est semi-calculable (tant que l'on n'a pas trouvé ces deux nombres, il faut essayer le nombre premier suivant). De plus, il a été montré que la calculabilité était identique dans tous les systèmes formels (logique mathématique, fonctions récursives, λ-calcul,...) et dans tous les langages de programmation.

Patrick Sallé, professeur à l'Institut national polytechnique de Toulouse, ENSEEIHT, 2, rue Charles Camichel, 31071 Toulouse Cedex.

LES LANGAGES FONCTIONNELS ET LOGIQUES

Programmation fonctionnelle et programmation logique ont en commun l'utilisation de concepts mathématiques au niveau de la définition des langages, mais diffèrent par la nature de ces concepts. Leur évolution s'oriente vers une intégration des deux styles de programmation. Leur importance vient de leur fondement logique et des outils de construction et de preuves qui leur sont associés.

FUNCTIONAL AND LOGICAL LANGUAGES - The combination of two programming styles, functional and logical, leads to the development of new languages with an improved expressivity and a logical foundation. Current research aims at the conception of frameworks for programming, validating and synthesizing software.

Didier Galmiche
Hélène Kirchner

La programmation fonctionnelle a pour composante de base la notion de fonction. Comme en mathématiques, les fonctions ont un domaine et un co-domaine et sont des entités manipulables. La notion de typage, dans un langage fonctionnel comme ML (voir encadré), prend en compte cette information et permet de classifier les termes.

Une fonction peut admettre des fonctions comme arguments et résultats; on dit qu'elle est alors d'ordre supérieur. On peut ainsi écrire des compositions de fonctions, des itérateurs sur des structures de données (comme l'itération d'une opération sur une liste), concevoir des programmes réutilisables (tel un tri paramétré par un ordre), mettre en évidence la structure d'un calcul d'ordre supérieur (par exemple, pour programmer des méthodes de démonstration).

La programmation logique, dont PROLOG est le précurseur, se fonde sur la logique du premier ordre et une procédure de déduction soigneusement contrôlée comme mécanisme d'exécution.

Le concept de contraintes a révolutionné la programmation logique des dix dernières années en lui apportant une plus grande puissance d'expression, une efficacité accrue et un vaste champ de recherche. Une contrainte est une formule qui caractérise un sous-ensemble du domaine de calcul. En PROLOG III, un programme est un ensemble de clauses assorties de contraintes qui permettent de réduire et maîtriser l'espace de recherche du problème. Les langages à contraintes sont largement utilisés dans des applications variées de recherche opérationnelle, de robotique, de vérification de circuits, de géométrie ou de calcul symbolique.

Intégrer les aspects fonctionnels et logiques

Intégrer dans un cadre unifié les aspects fonctionnels et logiques présente des avantages certains : en particulier, disposer de fonctions et prédicats dans un même langage et réduire l'espace de recherche par des calculs fonctionnels.

Les premières tentatives d'intégration ont consisté à implanter par exemple le système PROLOG en LISP et à appeler l'interpréteur LISP à partir d'un programme PROLOG, ou vice-versa. D'autres approches ont ensuite introduit des variables dites logiques dans un langage fonctionnel. Mais les voies les plus prometteuses pour intégrer prédicats et fonctions dans un même langage conduisent à combiner leurs mécanismes d'exécution. A titre d'exemple, λ-PROLOG introduit dans PROLOG les fonctions et l'ordre supérieur, autorise le polymorphisme, des variables de fonctions, et des buts contenant l'implication et la quantification universelle.

Pour développer des programmes, il est nécessaire d'adopter un style de programmation modulaire qui permet de décomposer un problème en sous-problèmes. La mise en œuvre de la modularité conduit aussi à la prise en compte de l'approche orientée objet et des notions de sous-type et d'héritage. Les contraintes, les fonctions d'ordre supérieur, la paramétrisation sont des outils fondamentaux pour modulariser des programmes et réutiliser des modules.

La multitude de propositions de langages logico-fonctionnels témoigne de l'intérêt d'une vision logique de la programmation. Formellement, un programme est un ensemble de formules d'un système logique, dont la syntaxe autorise suivant le degré d'expressivité recherché, fonctions, prédicats, types et polymorphisme. L'exécution est un processus de déduction efficace qui produit des conséquences du programme. Basée sur le fondement logique des langages, la programmation sûre du futur dépend aussi du développement des outils de preuve et d'aide à la construction de logiciels.

Preuves et programmes

Dans les langages fortement typés, la vérification des types fournit un premier outil de validation des programmes. Les déclarations de type et leur vérification à la compilation permettent de détecter un certain nombre d'erreurs fréquentes, dues à un mauvais ordre des arguments ou à une mauvaise orthographe d'un identificateur. Mais l'exploitation de la notion de typage va bien au-delà et conduit à l'étude de la théorie des types comme cadre de programmation, de validation et de synthèse de programmes corrects. Cette théorie se base sur la logique intuitionniste qui permet d'établir un parallèle entre types, formules et spécifications d'une part, et objets, preuves et programmes d'autre part. On obtient ainsi un cadre formel dans lequel on peut extraire des programmes fonctionnels corrects à partir de preuves de leur spécification. De plus, la logique linéaire, issue de travaux en logique intuitionniste, permet de prendre en compte les variables logiques, le non-déterminisme et un traitement logique du contrôle. Cette voie de recherche semble prometteuse pour aborder la construction automatisée de programmes corrects.

Puisqu'en programmation logique, les formules déduites en cours d'exécution sont, par nature, correctes, il reste à

PROLOG III

Le succès de la première version de PROLOG a conduit à un remaniement complet de ce langage. PROLOG III prend en compte des informations de type numérique, structurel ou booléen. En 1972, à Marseille, Alain Colmerauer mettait au point PROLOG, un langage de programmation d'un nouveau type spécialement conçu pour des applications d'intelligence artificielle telles que le traitement des langues naturelles ou la démonstration de théorèmes.

La rapide popularité acquise par ce langage (rappelons qu'il fut choisi par les Japonais pour leur projet de cinquième génération d'ordinateurs) amena d'une part une multiplication des PROLOG disponibles sur le marché, et d'autre part fit apparaître un certain nombre de carences du langage, liées en majeure partie aux traitements strictement symboliques effectués par PROLOG. La combinaison de ces deux phénomènes a introduit, ces dernières années, une surenchère des développements de PROLOG, à la fois sur le thème de l'efficacité et sur celui des fonctionnalités, par le biais d'une explosion de procédures externes ad hoc souvent fort éloignées des principes de base du langage.

En ce sens, PROLOG III ne s'inscrit pas comme une nouvelle version du langage PROLOG, mais présente un remaniement total (le noyau de l'interprète est cinquante fois plus volumineux que celui de PROLOG II) en incorporant au plus haut niveau de traitement la prise en compte d'informations de type numérique, structurel, ou booléen. Ce type de traitement (résolution de contraintes) ouvre la voie de la programmation logique à nombre de nouveaux domaines : diagnostic et simulation de circuits logiques, ordonnancement, planification, etc.

Si l'on revient sur la philosophie de programmation en PROLOG, on peut dire qu'une variable, en PROLOG comme en mathématiques, représente un objet inconnu. Cela est à opposer à la notion classique de variable en informatique, qui représente une adresse. Dans le même ordre de comparaisons, un programme écrit dans un langage procédural (FORTRAN, PASCAL, C, ...) tend à modifier les valeurs des variables pour calculer un résultat, alors que le but de l'exécution d'un programme PROLOG est de déterminer l'ensemble (éventuellement vide ou réduit à un élément) des valeurs possibles des variables auxquelles on s'intéresse.

Syntaxiquement, un programme en PROLOG III est constitué d'une suite de règles ; c'est pourquoi PROLOG est un langage déclaratif, par opposition aux programmes constitués de suites d'instructions. Les règles sont constituées d'une tête de règle, d'une suite de buts à exécuter, et d'un ensemble de contraintes portant sur certaines variables.

Ces caractéristiques de PROLOG III lui permettent de répondre facilement à des questions comme : combien faut-il de pigeons et de lapins pour totaliser ensemble 12 têtes et 34 pattes ? (réponse : 7 pigeons et 5 lapins). Ou encore : quels rectangles peut-on former avec neuf carrés dont les côtés sont tous différents ? (parmi les réponses : un rectangle 33 sur 32, un rectangle 69 sur 61).

La partie algèbre de Boole peut être utilisée par le petit problème de logique suivant, proposé par Georges Boole lui-même. Il s'agit de montrer que *quelque chose a toujours existé* à partir des cinq affirmations suivantes :

- quelque chose existe ;
- si quelque chose existe alors, soit quelque chose a toujours existé, soit les choses qui existent maintenant sont sorties du néant ;
- si quelque chose existe alors, soit ce quelque chose existe par la nécessité de sa propre nature, soit ce quelque chose existe par la volonté d'un autre être ;
- si quelque chose existe par la nécessité de sa propre nature, alors quelque chose a toujours existé ;
- si quelque chose existe par la volonté d'un autre être, alors l'hypothèse que les choses qui existent maintenant sont sorties du néant est fausse.

En associant chacune des propositions à une variable booléenne, et en posant comme contraintes les formules correspondant aux affirmations ci-dessus, PROLOG III parvient à une solution unique : quelque chose a toujours existé ; un autre problème est de savoir quoi.

Frédéric Benhamou, chercheur contractuel au CNRS, groupe Intelligence artificielle (URA 816 CNRS), Faculté des sciences de Luminy, case 901, 163, avenue de Luminy, 13288 Marseille Cedex 9.

LE LANGAGE ML

ML (*Meta Language*) était à l'origine un langage pour construire des preuves dans le système LCF (*Logic for Computable Functions*), puis est devenu un langage fonctionnel autonome avec ses fondements théoriques propres et différentes implantations (SML ou CAML). ML est un langage fortement typé, polymorphique, avec vérification automatique du typage et introduisant dans certaines versions le concept de module.

A titre d'exemple, considérons la fonction *filtre* qui applique un prédicat (une fonction à valeur booléenne) à une liste et retourne la sous-liste des éléments qui satisfont ce prédicat. La structure de liste est représentée par *[]* si la liste est vide et par *x: : L*$_1$ sinon (i.e. un élément *x* suivi d'une liste *L*$_1$). On définit *filtre* par :

```
letrec filtre = fun pred []          → []
                pred (x: : L1)       → if pred(x) then x: :filtre pred L1
                                       else filtre pred L1 ;;
```

Le type de *filtre* est déduit automatiquement de sa définition :
('a → bool) → 'a list → 'a list. Cette fonction est polymorphe car *'a* peut être n'importe quel type défini, et d'ordre supérieur car elle admet une fonction comme argument. Pour extraire d'une liste les éléments de taille au moins 4, on calcule par exemple :

```
filtre (fun x → taille x >= 4) ["CAML"; "est"; "le"; "ML"; "français"]
```

La définition récursive est utilisée et produit le résultat :
["CAML"; "français"] : string list, par un processus de calcul appelé réduction.

résoudre des questions du genre : le programme termine-t-il ? Le résultat trouvé est-il bien indépendant de l'exécution ? A-t-on trouvé toutes les réponses possibles à la requête posée ? Comment, dans un programme complexe composé de différents modules, peut-on combiner les propriétés précédentes vérifiées indépendamment pour chaque module ? Toutes ces recherches théoriques, en plein essor, devraient permettre l'extension de la programmation logico-fonctionnelle bien au-delà du formalisme de PROLOG ou de ML, et la construction d'environnements intégrés de programmation et de preuves.

Didier Galmiche, maître de conférences à l'Université de Nancy I, Centre de recherche en informatique de Nancy (CRIN) (URA 262 CNRS).

Hélène Kirchner, chargée de recherche au CNRS, CRIN (URA 262 CNRS), BP 239, 54506 Vandœuvre-lès-Nancy Cedex.

LANGAGES POUR LE PARALLÉLISME

Les machines parallèles, pour être efficaces, impliquent une programmation spécifique, qui peut être soit la traduction d'un programme séquentiel, soit un programme directement écrit en fonction des caractéristiques des machines parallèles.

Jean-Pierre Banâtre

LANGUAGE FOR PARALLEL COMPUTING - Parallel programming languages allow the description of systems of communicating sequential processes. Communication may be implemented by information sharing or by explicit message exchanges. Recent approaches based on more abstract models based on functional and logic paradigms are also investigated.

Les machines parallèles et les réseaux de processeurs ou de systèmes ont conduit à reconsidérer la conception des algorithmes et de leur expression. Il y a deux grandes approches de ce problème :

- Le parallélisme implicite. Des programmes séquentiels existants (écrits le plus souvent en FORTRAN) sont traduits automatiquement en programmes tirant profit des possibilités d'exécution parallèle offertes par la machine. Il est donc nécessaire de disposer de logiciels de traduction, appelés vectoriseurs. La qualité de la parallélisation dépend beaucoup plus de la forme du programme séquentiel que de la logique du problème traité.
- Le parallélisme explicite. Dans ce cas, on doit repenser les algorithmes en termes de processus coopérants, ce qui nécessite de nouveaux outils de programmation permettant de mettre en œuvre diverses formes de coopération entre calculs.

Deux éléments fondamentaux sont pris en compte dans les programmes parallèles : les calculs (appelés processus) qui transforment des données, et les communications qui permettent à ces processus d'échanger des informations.

Les interactions entre processus s'effectuent soit par partage d'informations communes, soit par échange explicite de messages.

Coopération par partage

La coopération par partage suppose que des processus puissent lire et mettre à jour des données que d'autres peuvent également consulter et modifier. Ces données doivent rester cohérentes (une écriture doit être terminée avant qu'une lecture ne puisse avoir lieu), il est donc nécessaire d'en contrôler l'accès. A cet effet, on utilise un mécanisme d'exclusion mutuelle permettant de construire des suites indivisibles d'instructions.

Historiquement, le premier outil permettant de résoudre le problème du partage de données fut le sémaphore. Un sémaphore est une variable entière positive ne pouvant être manipulée que par deux primitives appelées *P* et *V*. Ces deux primitives sont atomiques, c'est-à-dire que leurs exécutions s'excluent dans le temps. Les constructeurs de systèmes d'exploitation ont longtemps utilisé les sémaphores comme outils d'expression de la synchronisation entre processus ; cependant le degré d'abstraction fourni par cet outil s'est rapidement révélé trop faible pour la construction de programmes bien structurés. Diverses autres propositions ont vu le jour, la plus connue étant celle de moniteur.

Un moniteur est constitué d'une structure de données et d'un ensemble de procédures opérant sur ces données. La structure de données n'est pas directement accessible par les processus, seules les procédures peuvent être utilisées. Ces procédures s'exécutent en exclusion mutuelle, ce qui garantit un bon usage de la structure de données.

Coopération par échanges de messages

La coopération par échange explicite d'informations est exprimée à l'aide de deux primitives : envoyer (message, destinataire) et recevoir (message, expéditeur). Le destinataire et l'expéditeur sont des noms de processus, et le message une information typée. Suivant les langages, ces primitives obéissent à des règles de synchronisation différentes : la primitive d'envoi est bloquante ou non, suivant que l'on souhaite une communication synchrone (émetteur et récepteur travaillant en phase), ou asynchrone (l'émetteur peut envoyer un message *n* au récepteur sans information sur le traitement du message *n-1*). La primitive de réception est toujours bloquante : on ne traite pas une information avant de l'avoir reçue.

Le langage de référence dans la coopération par échange de messages porte le nom de CSP. Ce langage autorise la définition de processus communiquant par messages. L'échange d'informations est synchrone, c'est-à-dire que l'émetteur et le récepteur doivent être prêts à exécuter, l'un son opération d'envoi, l'autre son opération de réception, pour que la communication ait lieu (c'est le principe du rendez-vous).

La communication par appel de procédure peut être considérée comme une forme particulière de communication par messages. ADA propose un tel mécanisme de communication. Ce langage autorise la définition de processus (appelés tâches) et de procédures publiques (entrées) au sein du processus. Ces entrées peuvent être appelées par d'autres processus. Afin d'assurer une bonne synchronisation entre tâche appelante et tâche appelée, l'exécution d'une entrée est toujours sujette à acceptation par la tâche appelée.

Perspectives

On observe actuellement plusieurs approches nouvelles en programmation parallèle, en particulier la programmation descriptive parallèle et la programmation ensembliste.

La programmation descriptive parallèle s'appuie sur un modèle de programmation descriptive (fonctionnelle ou logique), auquel sont ajoutées des constructions de nature impérative permettant d'exprimer le parallélisme. Le plus souvent, on introduit une structure de données, appelée flot, que l'on peut considérer comme une suite à cardinal a priori infini, et une structure de contrôle opérant sur les flots. Cette structure de contrôle permet la production d'un "processus" consommateur dès qu'un nouvel élément est introduit dans le flot. En ce sens, on peut qualifier le calcul de "dirigé par les données".

La programmation ensembliste s'appuie sur un modèle de programmation dont la seule structure de données est l'ensemble sur lequel on applique des règles de transformations pouvant s'exécuter en parallèle. Le principe de synchronisation assure qu'une seule transformation peut s'opérer à un instant donné sur un élément particulier de l'ensemble.

Jean-Pierre Banâtre, professeur à l'INSA de Rennes, directeur de l'Institut de recherches en informatique et systèmes aléatoires IRISA (URA 227 CNRS), Campus de Beaulieu, 35042 Rennes Cedex.

LA PARALLÉLISATION AUTOMATIQUE

Le rôle d'un compilateur parallélíseur est de trouver, dans un programme rédigé dans un langage séquentiel classique tel FORTRAN, et sans intervention de l'auteur du programme, un maximum d'opérations pouvant être exécutées simultanément.

Pour que deux opérations puissent être exécutées en parallèle, il suffit que l'on puisse les intervertir sans changer l'effet du couple qu'elles constituent. Pour que cet échange soit possible, il suffit que ces deux opérations ne modifient aucune des cellules de mémoire qui leur sont communes. Le compilateur parallélíseur doit examiner sous cet angle tous les couples d'instructions du programme, en prenant en compte le fait que beaucoup d'instructions sont englobées par des boucles, et donc donnent lieu à un grand nombre d'opérations. Il enregistre les résultats de cette analyse sous la forme du graphe de dépendance ou GD, qui va servir de point de départ à la phase de synthèse. Le GD a un sommet par instruction, et chacun de ses arcs constitue un obstacle à la parallélisation du programme source. Le calcul exact du GD dans le cas général est une opération difficile et coûteuse. On se satisfait souvent de méthodes simplifiées, plus rapides, mais qui peuvent créer des arcs parasites et donc faire perdre du parallélisme.

A partir du GD, le compilateur synthétise un programme parallèle. Plusieurs paradigmes peuvent être utilisés à cette fin. Le plus simple consiste à identifier dans le GD des formes caractéristiques. Par exemple, une boucle parallèle est détectée en observant l'absence de certaines dépendances entre les instructions de son corps. Une boucle vectorielle est une boucle parallèle qui satisfait de plus à certaines conditions de complexité. Si cette approche ne suffit pas, ce qui est le cas général, le compilateur peut tenter de transformer le programme original dans le but de supprimer des obstacles à la parallélisation. Par exemple, la réutilisation trop systématique des cellules de la mémoire induit des dépendances qui peuvent être éliminées par expansion d'un scalaire en vecteur. Une boucle peut être rendue vectorielle en divisant ses instructions. A la pointe de la recherche, on trouve des compilateurs qui tentent de synthétiser directement le "meilleur" programme parallèle possible, sans passer par des étapes intermédiaires. La base de cette construction est un échéancier d'exécution du programme parallèle.

La compilation pour ordinateurs parallèles a encore des progrès à faire si on veut qu'elle se substitue à l'utilisation de langages spécialisés. Les méthodes simples peuvent s'appliquer à des programmes quelconques, mais ne dégagent que peu de parallélisme. Les méthodes plus puissantes sont coûteuses et ne peuvent s'appliquer qu'à des programmes très réguliers, tels les algorithmes du calcul matriciel. Il existe enfin une limite à ce que peut faire un compilateur : il est impossible de paralléliser un programme si les informations nécessaires à l'analyse n'y figurent pas, soit parce qu'elles font partie des présupposés implicites du programmeur, soit parce qu'elles sont cachées dans les données du programme. Franchir cette limite demandera sans aucun doute une révision profonde de nos habitudes et de nos langages de programmation.

Paul Feautrier, professeur à l'Université de Versailles-Saint Quentin, unité Méthodologie et architecture des systèmes informatiques (URA 818 CNRS), Centre universitaire de Versailles, 45, avenue des Etats-Unis, 78035 Versailles Cedex.

LA PROGRAMMATION VECTORIELLE

Le calcul vectoriel s'exprime aisément dans les programmes par des boucles dont toutes les itérations sont indépendantes et comportent exactement la même suite d'opérations. Les compilateurs, qui ont la charge de détecter ces boucles, sont ainsi devenus des vectoriseurs (voir l'encadré de P. Feautrier). De leur côté, les langages comportent maintenant des extensions qui permettent au programmeur d'écrire directement une instruction vectorielle allégeant par là-même la compilation. C'est ainsi qu'après beaucoup de tergiversations, la nouvelle norme FORTRAN (FORTRAN 90) vient de paraître, facilitant l'expression vectorielle des calculs.

La communauté scientifique a défini deux niveaux de procédures. Dans l'ensemble BLAS (*Basic Linear Algebra Subroutines*) ont été rassemblées les primitives de calcul vectoriel et matriciel. Il regroupe les opérations entre vecteurs, les opérations matrice × vecteur et les opérations matrice × matrice. Chaque constructeur doit en fournir une version adaptée à sa machine, version qui est en général écrite en assembleur pour en tirer le meilleur rendement. A un niveau supérieur, les différents algorithmes de résolution (systèmes linéaires, problèmes aux moindres carrés, problèmes aux valeurs propres) viennent d'être redéfinis dans la bibliothèque LAPACK écrite en FORTRAN standard, mais faisant systématiquement appel au niveau inférieur BLAS. Cette construction à deux étages assure une bonne efficacité et une portabilité à un moindre prix.

En fait, la bibliothèque LAPACK a été obtenue en reprenant un à un les algorithmes des bibliothèques précédentes LINPACK et EISPACK bien connues des usagers du calcul scientifique. Le travail a essentiellement consisté à définir des algorithmes qui manipulent des blocs de vecteurs plutôt que de simples vecteurs, introduisant par là-même l'emploi de la multiplication matricielle qui est l'opération la plus efficace. En effet, le produit matriciel s'exprime à l'aide d'une triple boucle emboîtée que l'on peut ordonner, partager à volonté. De plus, il a l'immense avantage d'entraîner un nombre de calculs très supérieur au nombre de données manipulées ($2n^3$ opérations flottantes pour $3n^2$ données dans le produit de deux matrices carrées d'ordre n), permettant ainsi l'utilisation efficace de mémorisations locales.

Il existe évidemment des opérations qui ne se prêtent pas facilement au calcul vectoriel car elles comportent des récurrences. Même dans ce cas un peu difficile, il existe parfois des algorithmes vectorisables mais au prix de calculs redondants. Par exemple, une récurrence linéaire du type $x_{k+1} = a_k \times x_k + b_k$ *pour* $k = 0, \ldots, n-1$, peut s'exécuter vectoriellement par l'algorithme dit de réduction cyclique avec environ 2,5 fois plus d'opérations que séquentiellement. Il faut donc que l'accélération du calcul vectoriel apporte un gain nettement supérieur pour que ce type d'algorithme soit considéré.

Pour améliorer encore l'adéquation entre l'algorithme et l'architecture de l'ordinateur visé, il faut remonter jusqu'à la formulation de la méthode de résolution elle-même. Par exemple, dans la résolution d'équations aux dérivées partielles, on peut être amené à choisir des méthodes de différences finies plutôt que d'éléments finis, lorsque l'on recherche une grande régularité dans les structures de données manipulées.

Jocelyne Erhel, chargée de recherche à l'INRIA, IRISA (URA 227 CNRS);
Bernard Philippe, directeur de recherche à l'INRIA, IRISA, Campus de Beaulieu, 35042 Rennes Cedex.

SYSTÈMES RÉACTIFS ET PROGRAMMES SYNCHRONES

Les programmes synchrones sont adaptés aux systèmes réactifs dont les systèmes d'informatique embarquée, dite "temps réel", sont le prototype. Leur principal avantage est de permettre de vérifier rigoureusement qu'ils satisfont à leurs spécifications.

SYNCHRONOUS PROGRAMMING OF REACTIVE SYSTEMS - In most realtime and automation applications, the pace of computation is imposed by the environment. Writing and validating programs that meet such requirements is particularly difficult. The novel class of synchronous languages, based on conceptually instantaneous instruction execution, offers new elegant and rigorous solutions.

Gérard Berry

Un système informatique réactif est chargé de maintenir une interaction constante avec son environnement, à un rythme imposé par cet environnement. Cette notion unificatrice récente recouvre un grand nombre d'applications développées jusqu'ici de façon disparate : contrôle de processus industriels, de robots, d'avions ou de moyens de transport, automatismes, protocoles de communication, traitement du signal, contrôleurs de circuits matériels, contrôleurs d'entrées-sorties ou d'interfaces homme-machine en programmation système. Son domaine privilégié – mais non exclusif – est celui de l'informatique embarquée dite "temps réel" où des ordinateurs sans écrans ni claviers contrôlent les objets dont nous nous servons tous les jours. Dans ce domaine, il est clair que la sécurité des programmes est particulièrement cruciale : le coût des bogues peut être gigantesque, parfois exprimable en vies humaines. Les exemples d'ennuis majeurs commencent malheureusement à se multiplier (accidents d'avions, écroulement de réseaux téléphoniques, etc.).

Une programmation délicate

Les méthodes utilisées classiquement pour programmer sont inadaptées au cas des systèmes réactifs, car elles ne savent pas concilier leurs deux exigences principales : le parallélisme d'expression, qui impose de décrire l'architecture des systèmes sous forme de modules autonomes communiquant entre eux, et le déterminisme comportemental, qui exprime qu'un même système placé dans les mêmes conditions doit toujours réagir de façon identique.

Les outils évolués que sont les langages de programmation parallèles classiques (ADA, MODULA, OCCAM) permettent le parallélisme d'expression, mais au détriment du déterminisme comportemental. Leurs processus communiquent par un système analogue au courrier postal. Un programme élémentaire comme "toutes les 60 secondes, signaler une minute" enverra une lettre à tous ses destinataires, qui la recevront lorsque la poste aura daigné la leur transmettre et qu'ils auront pris le temps de l'ouvrir, ce qui fait que la transmission de la minute pourra prendre un temps quelconque en théorie et en pratique. On imagine mal conduire correctement un avion avec un système de communication d'un tel type. Cette critique ne vise que l'application aux systèmes réactifs : les langages ci-dessus gardent toute leur valeur pour les systèmes distribués, dits interactifs, dans lesquels le rythme de l'interaction est choisi par les programmes eux-mêmes et non pas par l'environnement, comme par exemple les systèmes d'exploitation ou les algorithmes de calculs numériques sur machines parallèles.

A l'opposé exact, d'autres outils traditionnels comme les automates finis permettent de respecter le déterminisme mais ne supportent pas le parallélisme, ce qui rend l'écriture et surtout la lecture des programmes quasiment impossibles. En bref, disons que tous les outils classiques collent trop directement à la façon dont fonctionnent effectivement les ordinateurs et empêchent d'exprimer le problème à

LES LANGAGES SYNCHRONES

Il existe plusieurs langages synchrones actuellement en cours d'industrialisation et d'utilisation industrielle. ESTEREL, développé à l'Ecole des Mines et à l'INRIA Sophia-Antipolis, LUSTRE, développé à l'IMAG/LGI Grenoble et SIGNAL, développé à l'IRISA Rennes sont français et complémentaires dans leur style; ils sont respectivement industrialisés par CISI-Ingénierie et ILOG, VERILOG et TNI. Comme utilisateurs industriels, citons entre autres : Aérospatiale, Dassault Aviation, Merlin-Guérin, Renault, Siemens, Thomson, Toyota. Les Statecharts développés en Israël sont industrialisés aux USA et très utilisés en avionique et en transport ferroviaire. Deux langages de recherche, SML (Carnegie-Mellon) et ARGOS (IMAG/LGI Grenoble) ne sont pas encore industrialisés. A l'heure actuelle, la France est le leader du domaine.

Le développement des langages synchrones français a été subventionné par le Projet de Recherches Coordonné C3 du CNRS et du MRT et par l'action CAO-Automatique C2A du CNRS à laquelle participent de nombreux industriels. Dans le cadre de cette action, les équipes et industriels associés ont engagé une action de normalisation des codes fournis par leurs compilateurs respectifs qui permettra à terme de les rendre compatibles entre eux.

programmer suivant sa logique propre. Or, il est maintenant reconnu que la qualité stylistique et la rigueur des programmes sources sont un élément majeur de leur sécurité.

Les langages synchrones

Les langages synchrones apportent une vision radicalement nouvelle de la programmation réactive. Leur principe général est de supposer que toutes les interactions sont à temps nul, comme si l'on disposait d'ordinateurs infiniment rapides. On peut ainsi réconcilier parallélisme et déterminisme et obtenir un style de programmation parfaitement élégant et approprié aux problèmes. Pour reprendre notre analogie postale, les langages synchrones remplacent le courrier par la téléconférence.

Techniquement, notre hypothèse est appelée hypothèse de synchronisme parfait ; elle donne lieu à des développements mathématiques originaux. De fait, tous les langages synchrones actuels sont définis de façon complètement mathématique, car tous leurs auteurs et utilisateurs pensent que c'est la seule façon d'obtenir la sécurité requise.

Au premier abord, l'hypothèse de synchronisme parfait paraît absurde puisque aucun ordinateur ne peut calculer en temps nul. Mais les mathématiques du synchrone permettent d'effectuer des calculs formels très sophistiqués sur les programmes et de produire des codes objets extrêmement efficaces, ou même d'engendrer directement des circuits électroniques réagissant en quelques nanosecondes. En pratique, le synchronisme parfait est respecté si le temps de réponse du code objet – qui est très faible et prédictible – est inférieur au rythme imposé par l'environnement. La situation est la même que pour la mécanique newtonienne qui repose aussi sur une hypothèse d'interaction instantanée des corps : on sait que cette mécanique est fausse aux grandes vitesses, mais cela n'empêche en aucune façon son utilisation dans la vie de tous les jours.

L'existence de sémantiques mathématiques des langages synchrones permet de vérifier rigoureusement les propriétés des programmes et de démontrer, par exemple, que le contrôleur d'un ascenseur ne le fera jamais voyager la porte ouverte. Ces vérifications se font de façon automatique en utilisant entre autres des systèmes de manipulation formelle d'automates finis.

Gérard Berry, maître de recherches à l'Ecole des Mines de Paris, Centre de mathématiques appliquées, BP 207, 06904 Sophia-Antipolis Cedex.

LE GÉNIE LOGICIEL

Rendre les logiciels fiables et les mettre au point dans les délais prévus, telle est la tâche du génie logiciel. Des méthodes de plus en plus évoluées permettent d'approcher de ce but.

Marie-Claude Gaudel

Le génie logiciel est né en Europe, très exactement entre le 7 et le 11 octobre 1968, à Garmisch-Partenkirchen ; son parrain était l'OTAN. Il est rare de pouvoir donner les dates et lieux de naissance des domaines de recherche scientifique et technique. Mais le génie logiciel a ceci de spécifique qu'il a été défini, de toutes pièces, par un groupe de scientifiques pour répondre à un problème qui devenait de plus en plus évident : *d'une part, le logiciel n'est pas fiable, d'autre part, il est incroyablement difficile de réaliser dans des délais prévus des logiciels satisfaisant leurs cahiers des charges.*

"Le logiciel n'est pas fiable". Plusieurs logiciels erronés ont fait parler d'eux. La première sonde *Mariner* vers Vénus s'est perdue dans l'espace à cause d'une erreur dans un programme FORTRAN. En 1971, lors d'une expérience météorologique en France, 72 ballons contenant des instruments de mesure furent détruits tous d'un coup à cause d'un défaut dans le logiciel ... En 1981, un problème de logiciel retarda de deux jours la première mise en orbite de la navette spatiale, qui fut d'ailleurs lancée sans que l'on ait localisé exactement la cause du problème (mais les symptômes étaient bien délimités!). Dans la nuit du 15 au 16 décembre 1990, les abonnés de ATT de la côte Est des USA furent privés de tout appel longue distance à cause d'une réaction en chaîne dans le logiciel du réseau, due à un changement de version ...

Certains projets n'aboutissent jamais

"Il est incroyablement difficile de réaliser dans des délais prévus des logiciels satisfaisant leurs cahiers des charges". Ce fait est moins connu du grand public. Certains projets n'aboutissent jamais, ce fut le cas d'un compilateur PL1 chez Control Data dans les années 70. D'autres aboutissent avec des retards importants. Tout récemment EDF a dû remettre en cause la mise en service du nouveau système de contrôle-commande de ses centrales de 1400 mégawatts après plusieurs années d'efforts.

SOFTWARE ENGINEERING – Software engineering was born in 1968. Its aim is to develop correct and reliable software with control of costs and delays. Often programming accounts for less than 20% of the development effort. Specification, design, and validation are the main activities. Mathematical methods for developing and verifying software are being sought by researchers with some significant results.

Le projet sur les difficultés duquel on a le plus d'informations est le développement de l'OS-360 d'IBM ; c'est d'ailleurs tout à l'honneur d'IBM d'avoir accepté de discuter publiquement de ces difficultés. Il s'agit du système d'exploitation des ordinateurs de la gamme IBM 360. C'était un système innovant pour son époque. Or, malgré toute la compétence et la puissance économique d'IBM, il fut livré en retard ; il nécessitait plus de mémoire que prévu, le prix de revient dépassait de beaucoup les estimations, et les premières versions comportaient des erreurs.

Il faut toutefois éviter de sombrer dans un catastrophisme systématique. Comme toujours, on parle beaucoup de ce qui ne marche pas, et beaucoup moins de ce qui va bien ... Il y a des milliers de systèmes logiciels très respectables et qui fonctionnent tout à fait bien, sans faire parler d'eux. Les systèmes mentionnés ci-dessus sont presque tous des systèmes compliqués et qui, pour leur époque, comportaient beaucoup d'innovation. Cependant, tout utilisateur de système informatique s'est trouvé un jour ou l'autre confronté à un bogue : c'est-à-dire que dans un cas particulier, le système se comporte de manière aberrante.

Il faut aussi remettre les choses en perspective et se poser au moins deux questions : tout d'abord, un logiciel étant un produit manufacturé et complexe, est-il raisonnable d'en attendre une qualité totale ? On n'a pas de telles exigences dans les autres domaines technologiques. Après tout, les voitures tombent en panne, les circuits ont des défaillances. Par ailleurs, n'arrive-t-il pas que des projets autres que de développement de logiciels prennent du retard et dépassent leur budget ? Oui, bien sûr, mais il faut être hon- ▶

nête : dans le cas du logiciel, ces dépassements en temps et en coût peuvent être énormes (jusqu'à 300, 400 %).

Répondre à la demande

Il existe, enfin, un domaine où le logiciel pose un problème spécifique : il arrive que le logiciel ne corresponde pas à la demande. De même que la réalisation d'un objet complexe nécessite des schémas et des plans, la réalisation d'un logiciel important ne peut s'envisager sans spécification de ses caractéristiques. Comment exprimer ces spécifications ? Comment les soumettre au futur utilisateur et au futur réalisateur ? Dans ce domaine tout a dû être inventé, ou est en train d'être inventé. En effet, a priori, un logiciel, surtout en cours de développement, est invisible : on ne peut l'observer qu'en l'utilisant et ce type d'observation est souvent trop tardif et insuffisant pour comprendre ce qui se passe.

Aujourd'hui, on demande de plus en plus au logiciel : on met du logiciel dans les centraux téléphoniques, les centrales nucléaires, les fusées, les avions (Airbus A 320), le TGV et le RER, sans parler des banques ou de la bourse ... Les problèmes deviennent de plus en plus critiques. La taille des logiciels augmente : de l'ordre du demi-million d'instructions pour une expérience de physique des particules du CERN, du million pour un central téléphonique. Le plus grand projet de génie logiciel jamais réalisé est l'ensemble du contrôle au sol et en vol de la navette spatiale. Il comporte cinquante millions d'instructions, ce qui représente un effort de développement de plusieurs milliers d'hommes-année.

Or, on ne sait pas encore complètement maîtriser la fabrication de tels produits ; on ne sait pas bien évaluer le résultat d'une telle fabrication ; c'est un des paris scientifiques et technologiques actuels. Le génie logiciel a induit des problèmes spécifiques : la vérifiabilité, la modularité et l'encapsulation, l'abstraction, etc. Pour vérifier la construction d'un programme, il est utile d'avoir affaire à une programmation modulaire. Le concept de module n'est rien d'autre que celui de la pièce détachée ; il n'est pas naturel en informatique où les ordinateurs sont très centralisés. Il est essentiel qu'un module ne modifie jamais son voisin. Aussi est-on conduit à "encapsuler" les modules, c'est-à-dire empêcher que leurs données locales soient utilisées ou modifiées par d'autres modules. La méthode de base de la décomposition en modules est l'abstraction qui est basée sur le vieil adage : "diviser pour conquérir" ; il s'agit de maîtriser la complexité en ignorant des détails qui ne sont pas utiles ; les détails sont regroupés dans la partie locale du module, les aspects abstraits en sont l'interface.

La programmation : 15 à 20 % du temps

Il est important de noter que la programmation ne représente que de 15 à 20 % de l'activité totale de développement, la spécification et la conception environ 40 % de l'activité dans un projet bien conduit. La validation est du même ordre. De plus, la maintenance et l'évolution exigent souvent deux fois l'effort de développement initial.

La spécification définit ce que fait le logiciel, sans détailler comment il le fait. La conception introduit une décomposition en modules dont on ne définit que les caractéristiques et les interfaces. Depuis le début du génie logiciel, les informaticiens rêvent d'une spécification formelle qui garantirait le bon fonctionnement du logiciel. Les chercheurs ont, en conséquence, travaillé à l'élaboration de langages de spécification, non seulement descriptifs mais souvent exécutables en raison de leur formalité. On utilise pour cela des techniques voisines de celles de la démonstration automatique de théorèmes pour évaluer les fonctions décrites dans la spécification. Ces langages ne sont pas forcément universels. Certains formalismes sont bien adaptés à la description d'aspects spécifiques des systèmes informatiques : par exemple les réseaux de Petri pour la synchronisation, les types abstraits algébriques pour les types de données, la logique temporelle, etc. On peut donc imaginer avoir plusieurs spécifications de différents aspects d'un même système, comme on a une description de la carrosserie, du schéma électrique, du moteur, etc., pour une voiture. D'ailleurs, dans l'état actuel des choses, certains aspects ne sont pas spécifiables formellement. Ce sont essentiellement les aspects quantitatifs : performances, encombrements en mémoire, fiabilité, précision des calculs ...

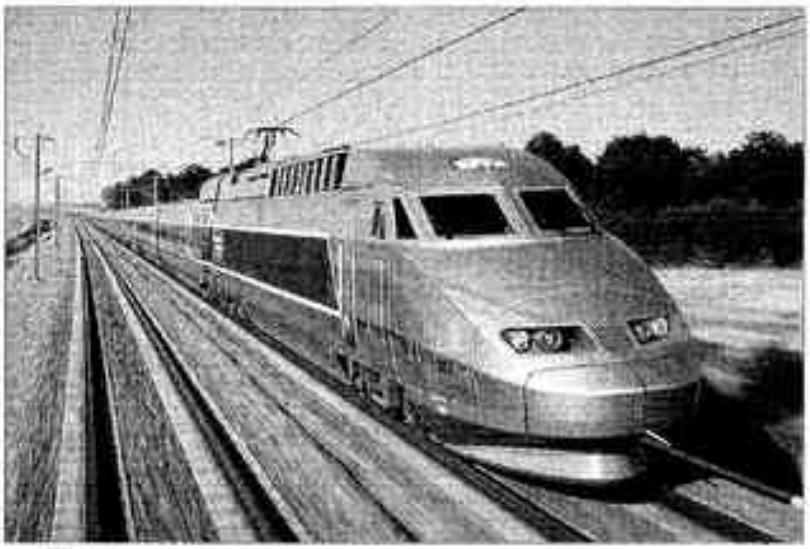
La validation

Pour s'assurer que le logiciel satisfait bien aux besoins, il faut le valider. Cette activité est complémentaire d'une vérification interne qui concerne chacun des modules.

Une des méthodes les plus couramment employées est la fabrication d'une maquette, c'est-à-dire d'un logiciel plus simple, réalisé rapidement à partir des spécifications, qui permet de se faire une idée de ce que sera le produit final, même s'il n'en assure pas toutes les fonctions. Il est intéressant de noter que dans un logiciel de grande taille, moins de 10 % des instructions concernent le but du système. Le reste assure les fonctions annexes de lecture, affichage, stockage des données, maintien de la cohérence, etc.

Le maquettage se pratique depuis longtemps. APL a été très utilisé en son temps. La nature des problèmes traités étant de plus en plus symbolique, on s'oriente maintenant vers l'utilisation de langages comme PROLOG ou des langages à objets. ADA offre également des possibilités intéressantes pour le maquettage, du fait de la séparation des modules du langage en une partie interface et une partie corps.

Les spécifications formelles ont ouvert de nouvelles perspectives de validation. En effet, on peut faire beaucoup de choses à partir d'une spécification formelle : par exemple, des preuves que certaines propriétés sont toujours fausses ; on peut



Rendre les logiciels fiables, un objectif du génie logiciel qui intéresse au plus haut point la SNCF pour ses TGV (© SNCF - CAV - J.-M. Taberlet).

aussi, quand le langage de spécification le permet, exécuter la spécification, la tester. Ces activités aident à se convaincre que la spécification est "satisfaisante" avant de commencer le développement. Le fait que la spécification soit formelle permet de faciliter l'activité de validation, mais il n'élimine pas totalement le risque que les besoins des utilisateurs aient été mal formulés ou mal compris.

La vérification

La vérification confronte un logiciel à sa spécification. L'idéal serait de faire une preuve mathématique de leur correspondance. De telles preuves se font en utilisant des systèmes formels comme la logique de Hoare ou les plus faibles préconditions de Dijkstra qui sont connus depuis longtemps. La preuve de programme est restée longtemps peu appliquée, mais les techniques de démonstration automatique de théorèmes évoluent vite et profitent de certains résultats de l'intelligence artificielle. Il en est de même des systèmes de calcul symbolique (Macsyma, Reduce), qui permettent de raisonner sur un large sous-ensemble des mathématiques. De plus, les performances des machines permettent d'envisager de conduire ou de vérifier des preuves longues.

Des exemples récents et très convaincants de projets comportant des spécifications formelles et des preuves ont été publiés : la vérification formelle d'un algorithme tolérant aux fautes de synchronisation d'horloges dans un système distribué ; la preuve de correction des programmes du système SACEM qui contrôle la survitesse sur la ligne A du RER (21 000 lignes de Modula, partiellement embarquées) ; la vérification de la conception d'un système d'exploitation tolérant aux fautes pour un système de contrôle de vol ; des travaux sont menés et appliqués chez Merlin-Guérin autour de la méthode SAGA, etc. Toutes ces preuves s'avèrent être longues, fastidieuses, et il y a un consensus général sur le besoin de systèmes pour vérifier et assister ces activités. De tels systèmes existent maintenant (Boyer Moore, LHDM, Gypsy, HOL, MALPAS...).

Mais aucun système ne peut éviter la phase de test. Le test statique n'est rien d'autre qu'une analyse détaillée du programme, sans qu'il soit exécuté. Le test dynamique consiste à exécuter le programme sur un sous-ensemble de données. Le choix de ces données peut être structurel (basé sur la structure du programme), fonctionnel (dépendant des spécifications) ou aléatoire. Toutes posent le problème de l'oracle, c'est-à-dire la décision du succès ou de l'échec du test. Pour vérifier que le programme fonctionne bien sur une donnée précise, il faut connaître au préalable la réponse qu'il doit donner.

Les outils

Les recherches sur les environnements de développement et de maintenance sont pour l'essentiel consacrées au problème de l'intégration des outils. Ceux-ci sont en effet rarement compatibles, et il est parfois plus long ou plus difficile de mettre un programme (ou une spécification, ou un autre document relatif à un logiciel) produit par un outil n° 1, sous une forme acceptable par un outil n° 2, que de se servir des outils eux-mêmes. La solution à ce problème passe par la définition de *bases de projet* permettant de gérer de manière uniforme les nombreuses informations associées au développement d'un logiciel. Les modèles de bases de données classiques ne sont pas bien adaptés à ce problème : en effet, ils permettent de gérer un très grand nombre d'objets d'une même structure simple, alors que l'on doit gérer ici des objets de structures complexes et très variées, avec relativement peu d'objets de chaque type. La définition de modèles bien adaptés conditionne l'obtention de véritables *ateliers intégrés* de génie logiciel. La conception et la réalisation de tels ateliers demande des moyens importants et pose encore des problèmes scientifiques et techniques difficiles : il n'est pas étonnant que, dans le contexte européen, ces projets se déroulent dans le cadre d'ESPRIT ou d'EUREKA (structure d'accueil PCTE-Emeraude, EAST, ESF). Ces projets européens sont d'ailleurs en pointe dans le domaine.

En France, on parle d'AGL (Ateliers de Génie Logiciels), aux USA, de CASE (*Computer Aided Software Engineering*), en Angleterre, d'IPSE (*Integrated Platform for Software Engineering*), ce qui montre l'intérêt énorme pour de tels systèmes, à la hauteur des enjeux économiques.

Il faut noter que ces systèmes sont essentiellement faits de logiciels. Ce n'est pas là le moindre paradoxe : il s'avère que pour développer les outils qui permettront de résoudre les problèmes du génie logiciel, il faut développer des logiciels complexes, donc avoir résolu ces problèmes !

Marie-Claude Gaudel, professeur à l'Université Paris-Sud, Laboratoire de recherche en informatique (URA 410 CNRS), Bât. 490, 91405 Orsay Cedex.

LES ENVIRONNEMENTS DE PROGRAMMATION

Un bon environnement de programmation facilite le travail du programmeur et en améliore la qualité. Il présente à l'utilisateur des événements très concrets sous une forme abstraite.

Gilles Kahn

PROGRAMMING ENVIRONMENTS – The tools offered to engineers to construct software are evaluated on the basis of practical criteria. Very many factors need to be taken into account. Inserting part of program into an existing application raises specific problems relating to context. Programming environments for large systems are closely related to symbolic computing.

Programmer, c'est construire, en partant d'une description informelle d'un problème, des documents extrêmement précis qui permettent à l'ordinateur de le résoudre.

L'*environnement de programmation* est l'ensemble des outils qui aident l'ingénieur à programmer. Sa qualité se juge sur des critères pratiques : l'ingénieur est-il efficace ? est-il déchargé de tâches subalternes ? peut-on avoir confiance dans le résultat ? en cas d'erreur, est-on aidé dans le diagnostic ? en cas de modification dans les spécifications, les changements sont-ils facilités ?

Le plus simple des environnements de programmation est limité à un *éditeur de texte* et un *compilateur*. L'éditeur de texte sert à construire tous les documents utilisés par le programmeur. Le compilateur effectue certaines vérifications de cohérence et traduit les programmes en langage machine. Lorsque des anomalies se produisent, il est difficile de les mettre en relation avec le texte du programme. Les faiblesses d'un tel environnement sont évidentes : aucun support pour la composition de documents spécialisés (programmes et spécifications), pas de communication entre les divers outils, processus de développement rigide.

Des améliorations considérables sont intervenues dans les dix dernières années : les éditeurs de texte peuvent avoir des modes spécialisés ; les compilateurs émettent des messages qui sont associés visuellement avec le texte du programme ; la mise au point est interactive et graphique. L'un des objectifs de la recherche est d'analyser l'architecture nécessaire pour construire un environnement de qualité spécialisé pour un nouveau langage de programmation. Ce travail implique de comprendre la *sémantique* des langages de programmation et la nature des *protocoles* de communication entre outils : des événements très concrets doivent remonter une longue chaîne de transformations pour être présentés à l'utilisateur dans les termes abstraits qui sont les siens.

Industriels et chercheurs mettent aujourd'hui l'accent sur des outils menant à une construction plus systématique. Certaines analyses faites par le compilateur par exemple peuvent avoir un intérêt pour le programmeur, si toutefois leurs résultats sont présentés de manière appropriée. Les programmeurs ont l'habitude de prendre en considération des questions d'efficacité trop tôt, ce qui rend les programmes obscurs et difficiles à maintenir. Un ensemble d'outils d'analyse, d'instrumentation et de transformations mécaniques de programmes permet d'effectuer l'optimisation de manière rationnelle. La construction d'un programme s'éloigne ainsi progressivement de la conception d'un texte pour se rapprocher de la construction d'une formule susceptible de transformations et d'évaluations diverses.

De plus, il est exceptionnel de construire un fragment de programme totalement autonome : un programme vient presque toujours s'insérer dans un contexte, ensemble de programmes et de types de données préexistant. Il est nécessaire de s'orienter dans un tel ensemble, de s'y insérer de manière cohérente et de réutiliser les éléments qui y sont présents. L'environnement de programmation doit fournir les outils – menus, recherche de composants, accès aux spécifications – qui rendent systématique et naturelle l'exploitation du contexte. Une interface graphique moderne est indispensable.

Cette vision globale de l'activité de programmation fait l'objet des recherches actuelles. Elle rapproche les environnements de programmation des grands systèmes de calcul symbolique et le développement des programmes d'un problème qui est loin d'être résolu de manière satisfaisante : la construction de systèmes interactifs aidant à faire des démonstrations mathématiques.

Gilles Kahn, directeur de recherche à l'INRIA, 2004, route des Lucioles, Sophia-Antipolis, 06565 Valbonne.

Centaur : un environnement de programmation interactif (logiciel développé à l'INRIA) (Cliché INRIA - A. Estublier).

LANGAGES À OBJETS

Le fonctionnement d'un programme par objets ne se déroule pas comme un algorithme. Il ressemble plutôt à l'animation d'un modèle réduit qui simule la réalité.

Jean-François Perrot

OBJECT-ORIENTED LANGUAGES - Objects are gaining general acceptance. They induce a new way of programming by putting together structure and behavior, as specified by classes. Classes are approximations of concepts, and participate in the inheritance relation.

Depuis une dizaine d'années, la programmation par objets a gagné une réputation enviable dans le monde de l'informatique : de l'aveu général, elle facilite grandement la conception et l'écriture de logiciels de qualité. Dans le détail, le sens donné au qualificatif "par objets", traduction de l'anglais *object-oriented*, varie suivant la spécialité de l'interlocuteur.

L'idée de base est de considérer le fonctionnement du programme non pas comme le déroulement d'un algorithme, mais comme l'animation d'un modèle réduit. Ce modèle est formé d'objets informatiques qui reproduisent ceux du monde réel. Par objet, on entend une entité "concrète", "qui est là", conformément à l'étymologie, par opposition à une fonction ou à une procédure. En termes informatiques, cela signifie une entité permanente, qui existe indépendamment du nom qu'on lui donne, et qui est susceptible d'interagir avec d'autres objets.

Chaque objet composant le modèle peut être complexe. On souhaite pouvoir "oublier" cette complexité, en ne considérant que les interactions de l'objet avec ses congénères. Bien entendu, la complexité réapparaît ailleurs, au niveau de la conception de l'objet.

Structure et comportement

Le monde réel est évidemment beaucoup trop compliqué pour qu'on puisse en représenter, même un fragment, avec une fidélité totale. On ne représentera donc qu'une approximation, laquelle ne sera valable que vis-à-vis de l'utilisation qui en sera faite. La programmation par objets propose de définir cette approximation en termes de structure (l'objet "est fait comme ça") et de comportement (l'objet "sait faire ça").

La structure se formalise par un ensemble d'attributs qui sont eux-mêmes d'autres objets, et le comportement par une batterie de procédures. Les attributs, comme les procédures, portent des noms.

L'ensemble des valeurs des attributs matérialise l'état de l'objet, état qui peut varier au cours du temps. Le comportement est obtenu en faisant exécuter les différentes procédures, désignées par leur nom. Naturellement, le comportement de l'objet à un moment donné dépend de son état.

La spécification complète de l'objet, structure et comportement, se ramène à une entité textuelle unique composée, d'une part, de la liste des noms d'attributs et, d'autre part, des textes des procédures assortis de leurs noms. Cette entité s'appelle la classe de l'objet. Elle définit une infinité d'objets potentiels isomorphes, ses instances.

Définition des classes et création d'instances

La construction d'un modèle réduit se fait en deux étapes : définition des différentes classes d'objets qui doivent y entrer, puis création des instances dont l'ensemble compose le modèle. Le modèle une fois construit est activé pour lancer la simulation et obtenir les résultats désirés.

Soulignons qu'à la différence d'une maquette matérielle, le fonctionnement d'un tel modèle peut parfaitement provoquer la création d'objets qui n'étaient pas présents au départ (et leur destruction), souvent en très grand nombre, ce qui pose des problèmes de gestion de la mémoire de mieux en mieux résolus.

Le tandem classe/instance, très simple dans son principe, fournit une réalisation informatique du couple concept abstrait et instance actualisant ce concept. Conjugué avec l'approche simulatoire, il s'avère d'une redoutable efficacité. En outre, le plus souvent la plupart des classes utiles à un projet peuvent être extraites d'une bibliothèque de classes préexistantes, soit directement, soit via le mécanisme d'héritage.

Pour qu'une classe A soit plus générale qu'une classe B, il suffit que tous les attributs et toutes les procédures que définit A se retrouvent dans B (mais non l'inverse). On dit alors que B est une sous-classe de A et on convient de construire B en n'écrivant que les attributs et procédures supplémentaires par rapport à A. Cette relation de généralisation spécialisée s'appelle relation d'héritage. Elle permet d'organiser les classes d'un système en un graphe d'héritage. Elle fonde une technique de développement par raffinements successifs, partant de classes générales assez simples, pour arriver à des classes particulières, plus complexes. On peut ainsi récupérer de manière indirecte des classes préexistantes. Les classes jouent donc un double rôle, elles sont à la fois génératrices de leurs instances et mères de leurs sous-classes. L'emploi judicieux de l'héritage est la clé du succès en programmation par objets.

Chaque langage de programmation par objets est accompagné d'une bibliothèque de classes plus ou moins bien fournie. Ces langages sont aujourd'hui nombreux : après le pionnier SIMULA (né en 1967, toujours vert) sont venus SMALLTALK-80 qui a lancé la vogue des objets, suivi par OBJECTIVE-C (1984), le poids lourd C++ (1985), EIFFEL (1986), le COMMON LISP OBJECT SYSTEM (CLOS, 1987, héritier des célèbres FLAVORS de la machine LISP, 1980), et d'innombrables systèmes d'intelligence artificielle.

Jean-François Perrot, professeur à l'Université Pierre et Marie Curie, Laboratoire formes et intelligence artificielle LAFORIA (URA 1095 CNRS), Boîte 169, 4, place Jussieu, 75252 Paris Cedex 05.

PREUVES ET CONSTRUCTIONS DE PROGRAMMES

Les utilisateurs réclament de plus en plus de programmes certifiés corrects, c'est-à-dire présentant une garantie contre les erreurs de comportement. Une méthode de choix pour les construire consiste à traduire des démonstrations mathématiques en programme informatique.

Gérard Huet
Christine Paulin-Mohring

PROOF AND PROGRAM CONSTRUCTION - Typed functional languages are the natural framework for translating mathematical demonstrations into computer programs. The programs that arise from this operation are automatically correct and documented. Furthermore, they are easy to check. Indeed, any proof of a proposition which does not involve the principle of the excluded middle can naturally be translated into a computer program.

Un programme informatique idéal doit faire intervenir trois composantes : les spécifications, le code et la documentation. Les spécifications décrivent ce que doit faire le programme. Le code permet à une machine d'exécuter la tâche. Il doit être correct, c'est-à-dire répondre exactement et dans tous les cas aux spécifications. La documentation, qui apparaît comme un commentaire, permet de comprendre la structure du programme et de la modifier sans en altérer la correction.

Dans la pratique on est souvent loin de cet idéal, la plupart des programmes sont incomplètement spécifiés, erronés et difficiles à faire évoluer. Une méthode prometteuse consiste à transcrire en langage informatique une démonstration mathématique. Nous allons expliquer cette méthodologie.

Les preuves vues comme programmes

La spécification du programme peut être représentée par un énoncé mathématique. Dans le cas simple où le programme doit transformer une entrée x vérifiant une propriété $P(x)$ en une sortie y telle qu'une certaine relation $R(x,y)$ entre x et y soit réalisée, la spécification sera l'énoncé : pour tout x tel que $P(x)$, il existe y tel que $R(x,y)$.

Si prouver un théorème est une tâche qui requiert de l'intelligence, par contre vérifier qu'une démonstration suffisamment détaillée est correcte est une tâche totalement mécanisable. Il est tout à fait possible de formaliser une preuve mathématique dans un langage qui permet à un ordinateur de la vérifier.

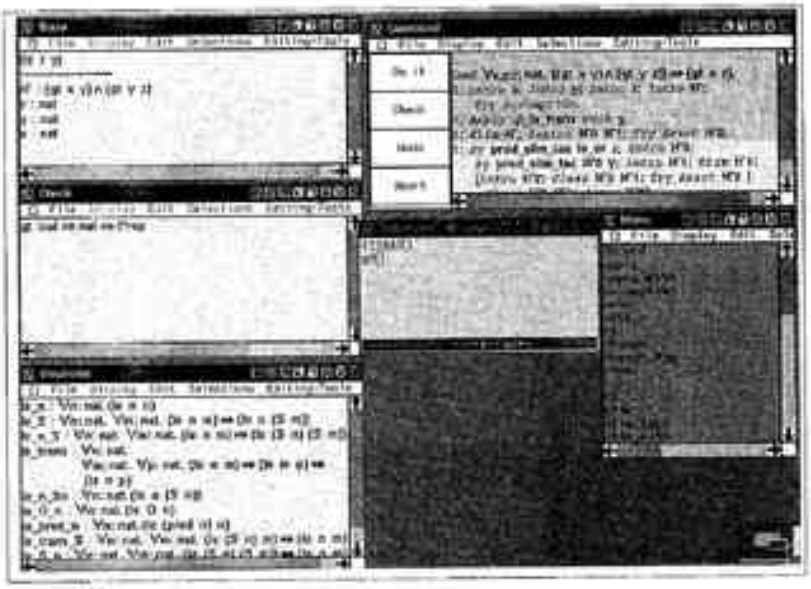
L'intérêt de cette approche est qu'une preuve peut être automatiquement transformée en un programme correct, c'est-à-dire qui transformera tout x vérifiant $P(x)$ en une sortie y telle que $R(x,y)$ soit vérifié. Dès 1942, le logicien Kleene a mis en évidence la traduction d'une preuve intuitionniste (sans utilisation du principe du tiers exclu) en un procédé calculable.

La preuve est alors vue comme un programme commenté qui intègre toute l'information nécessaire permettant de vérifier mécaniquement sa correction.

A propos du tiers exclu

L'axiome du tiers exclu permet, pour n'importe quelle propriété A, de raisonner par cas en supposant que soit A est vrai, soit A est faux. Pourquoi s'interdire a priori l'utilisation de ce principe ? S'il est naturel de raisonner par cas "si A alors ...sinon...", il n'est pas évident d'interpréter ce schéma de preuve comme une construction de programme. Si A est de la forme $n = m$, alors on s'attend à ce que la construction de programme "si $n = m$ alors...sinon..." soit une primitive du langage de programmation. Si A est plus compliqué, par exemple "n appartient à un ensemble S", alors il est probable que le programmeur ait explicitement à programmer une procédure qui répondra si oui ou non un objet n appartient à un ensemble S. La spécification d'un tel programme pourra être la formule "pour tout n, n appartient à S ou bien n n'appartient pas à S". Mais il n'est pas sûr qu'une telle procédure existe toujours ou bien soit connue. On voit donc que ce principe du tiers exclu n'est pas du tout une évidence d'un point de vue construction de programmes.

Un exemple d'utilisation non justifiée du tiers exclu est le suivant. Il s'agit de prouver l'existence de deux nombres a et b tels que a et b sont irrationnels mais a^b est rationnel. La preuve mathé-



Développement de preuves dans le système Coq utilisant l'interface réalisée avec CentaurI par Y. Bertot à l'INRIA, Sophia Antipolis.

matique suivante est tout à fait correcte :
- si $\sqrt{2}^{\sqrt{2}}$ est rationnel, alors comme $\sqrt{2}$ est irrationnel, la solution $a = \sqrt{2}$, $b = \sqrt{2}$ convient;
- si $\sqrt{2}^{\sqrt{2}}$ est irrationnel, alors comme $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = 2$ est rationnel, la solution $a = \sqrt{2}^{\sqrt{2}}$, $b = \sqrt{2}$ convient.

Cette preuve ne contient pas l'information nécessaire permettant de construire effectivement un couple (a,b) répondant à la spécification.

Cependant, il est possible de construire des programmes à partir de programmes utilisant le tiers exclu, pourvu que la spécification de tels programmes soit assez simple. Des développement récents ont même montré que l'on pouvait directement donner un sens calculatoire à l'utilisation du principe du tiers exclu. Cependant, il faut pour cela faire intervenir des opérations qui ne sont plus "fonctionnelles", mais utilisent des "sauts" (par exemple des exceptions) dans le mécanisme d'évaluation du programme.

Une documentation automatique

La transformation d'une démonstration en programme informatique, quand elle est possible, conduit à un programme qui est automatiquement correct et documenté, puisque la succession de ses parties correspond à une succession de propositions logiques qui peuvent être explicitées. Nous nous trouvons donc en principe en présence du programme idéal. Dans la pratique, il existe cependant des limitations. Les programmes construits à partir des démonstrations mathématiques sont en général plus lourds et plus lents que les programmes classiques. Heureusement, les progrès en puissance et rapidité des machines modernes sont tels que cette limitation n'a plus la même importance que dans le passé.

Les types fonctionnels

Les types sont une méthode simple d'assurer une vérification mécanique partielle de la correction des programmes. Des types simples (le type "nombre entier" ou bien le type "des couples de deux entiers") sont utilisés dans les langages de programmation usuels. Le type d'une expression de programme peut être calculé mécaniquement ; il décrit le comportement du programme. Les types servent à restreindre les domaines d'application des méthodes et permettent ainsi la détection d'erreurs de programmation de manière statique (c'est-à-dire avant l'exécution des programmes). Le type est calculé suivant l'expression du programme, mais ce qui nous intéresse est qu'il décrit une propriété des résultats de l'exécution du programme. Cette propriété de conservation du type par réduction de l'objet semble évidente, mais elle est cependant mise en défaut dans bon nombre de langages usuels qui autorisent des manipulations telles que les affectations ou l'utilisation de pointeurs dont le comportement logique n'est pas clair.

Les langages fonctionnels typés (tels que ceux de la famille ML) sont d'autant plus intéressants qu'ils sont le cadre naturel de la traduction en programme informatique d'une preuve mathématique : il existe une correspondance directe entre le formalisme de la déduction naturelle et celui d'un langage fonctionnel typé (isomorphisme de Curry-Howard). Il est donc possible de transformer mécaniquement une preuve mathématique en un programme fonctionnel. Les types deviennent alors de véritables spécifications.

Les langages de l'avenir

L'analogie entre preuve et programme est particulièrement bien adaptée aux programmes fonctionnels, mais ceux-ci ne sont pas encore très utilisés dans des systèmes réels. Cependant, ils offrent une plus grande sécurité de programmation qui peut les faire préférer comme langage de développement de programmes, quitte à utiliser ensuite des traducteurs vers des langages de plus bas niveau pour des raisons d'efficacité et de portabilité.

Au-delà des aspects mathématiques de ces investigations qui procèdent de développements récents de la théorie de la démonstration (une branche de la logique mathématique), la mise en œuvre informatique de ces méthodes pose de nombreux problèmes d'ingénierie : conception et réalisation de systèmes de preuves interactifs suffisament conviviaux, de langages de programmation dont la sémantique est adaptée à la logique du système de preuves, d'environnements de construction de programmes permettant la mise au point de systèmes assemblant des modules produits mécaniquement à partir de preuves de leur spécifications, avec des modules développés par des méthodes traditionnelles. Un effort de recherche important sur ces thèmes rassemble de nombreuses équipes européennes au sein d'un consortium (*Esprit Basic Research Action* : TYPES) comprenant des logiciens et des informaticiens. Les industriels sécuritaires (transports, avionique, électronique,...) commencent à s'intéresser à l'utilisation en vraie grandeur des prototypes issus de la recherche. Progressivement, les outils de génie logiciel intégreront ces nouvelles techniques dans des environnements de développement de programmes plus sûrs et plus robustes.

Gérard Huet, directeur de recherche à l'INRIA, BP 105, 78153 Le Chesnay Cedex.
Christine Paulin-Mohring, chargée de recherche au CNRS, Ecole normale supérieure de Lyon, Laboratoire d'informatique du parallélisme-IMAG (URA 1398 CNRS), 46, allée d'Italie, 69364 Lyon Cedex 07.

LA VÉRIFICATION AUTOMATISÉE DES SYSTÈMES

Des méthodes rigoureuses de vérification du fonctionnement des systèmes sont aujourd'hui nécessaires. Elles impliquent la mise au point de formalismes précis des spécifications ainsi que des méthodes et outils adaptés.

Joseph Sifakis

Le développement de systèmes et logiciels fiables nécessite l'utilisation de méthodes de validation des choix des concepteurs, par rapport à des spécifications exprimant le service attendu. Par vérification nous entendons la comparaison d'un système souvent décrit par un programme, à des spécifications.

Les premiers résultats sur la vérification concernent les programmes séquentiels dont le comportement peut être spécifié par une relation entre l'état initial (la valeur des données) et l'état final (le résultat du calcul). Pour l'expression des spécifications de ces programmes, on distingue deux sortes de propriétés. La première, dite correction faible, signifie que le programme réalise une relation donnée d'entrée/sortie dans le cas où il se termine. La seconde, dite correction forte, exige en plus que le programme se termine.

La spécification dans les systèmes parallèles

Les systèmes parallèles imposent un autre point de vue. Ils sont composés d'agents qui coopèrent pour réaliser un comportement global. Les systèmes parallèles et leurs composants sont donc essentiellement réactifs, c'est-à-dire que leur rôle est de maintenir une interaction avec leur environnement. Il n'est plus possible de les considérer comme corrects en comparant état initial et état final. C'est leur comportement même qui doit être correct. Dans ces conditions, le problème de la spécification devient délicat, car il implique le choix et la définition de ce qui doit être observé, ainsi que la vérification de la cohérence et du caractère complet de la description qui en est faite. Selon le type de langage de spécification utilisé, on distingue deux approches. Dans la première, une spécification est un comportement observé à un certain niveau d'abstraction : le système peut prendre un ensemble d'états reliés par une relation de transition. Dans la deuxième approche, la spécification est un ensemble de propriétés globales de comportement du système telle que l'absence de blocage, l'exclusion mutuelle ou l'équité. Dans ce cas, les spécifications sont exprimées dans des formalismes basés sur une logique adaptée. Il est généralement admis que les deux approches de spécifications sont complémentaires et nécessaires.

Méthode déductive ou passage par des modèles ?

On distingue par ailleurs deux catégories de méthodes de vérification. La première consiste à définir un système déductif (un ensemble d'axiomes et de règles d'inférence) décrivant la relation voulue entre le programme et ses spécifications. Vérifier se ramène alors à une preuve dans ce système déductif. La seconde consiste à générer, à partir du programme à vérifier, un modèle qui représente sous forme d'un système de transitions (relation binaire sur les états) des aspects essentiels du comportement du programme. Le modèle correspond donc à une abstraction du programme dont le choix dépend en général des propriétés à vérifier. La méthode de vérification consiste à comparer les spécifications à ce modèle.

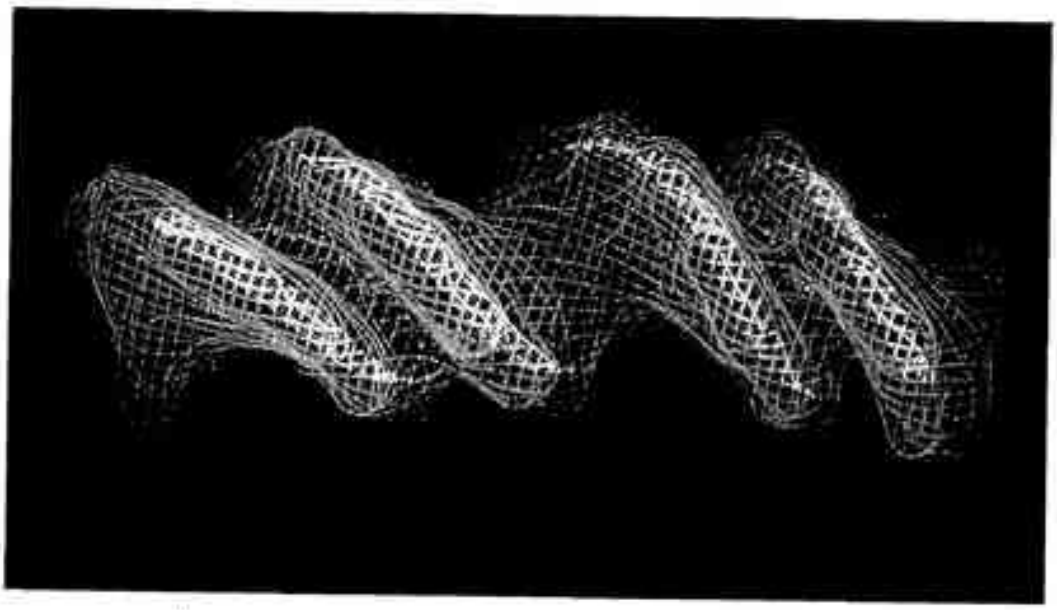
Quelle que soit l'approche suivie, la mise en œuvre des méthodes de vérification se heurte à des problèmes d'indécidabilité quand le système peut prendre une infinité d'états. Il est donc impossible de définir dans ce cas des méthodes générales de vérification automatique. Seules des méthodes interactives peuvent être envisagées. Leur efficacité dépend fortement des compétences de l'utilisateur.

AUTOMATED SYSTEM VERIFICATION – Two approaches to the specification and verification of concurrent and reactive systems are presented. One approach uses a deductive system to establish the fact that a program meets its specifications. The other consists in translating the program to be verified into a model which is compared against its specifications.

Heureusement, il existe des domaines d'applications tels que les systèmes de commande temps réel, le matériel et les protocoles de communication, où les systèmes peuvent être représentés par des modèles ayant un nombre fini d'états. La vérification automatisé de ce type de système devient donc possible en théorie, ce qui ne veut pas dire qu'elle le soit toujours en pratique pour des systèmes complexes. Les résultats expérimentaux concernant l'application de ces méthodes sont tout à fait encourageants. Les outils de vérification existants permettent de vérifier automatiquement des systèmes de complexité moyenne (modèle de quelques millions d'états ou programme d'une centaine de variables booléennes). L'évolution rapide du savoir-faire laisse prévoir que, dans un avenir très proche, des outils de vérification automatiques ou interactifs seront intégrés dans des environnements pour la conception de systèmes temps réel ou de matériel.

Joseph Sifakis, directeur de recherche au CNRS, Laboratoire de génie informatique de Grenoble (URA 398 CNRS), LGI/IMAG, BP 53X, 38041 Grenoble Cedex.

Informatique théorique et mathématiques



Brin d'ADN (© CNRS-LMC, Ph. Plaily). Certains algorithmes s'appliquent aussi bien à l'analyse des chaînes moléculaires qu'au traitement des chaînes de caractères.

L'informatique et les mathématiques sont deux disciplines distinctes mais qui ne sont pas indépendantes. Même des problèmes aussi simples que les opérations arithmétiques posent des problèmes mathématiques très difficiles, voire non résolus. Ainsi, l'algorithme de multiplication rapide de deux entiers, découvert par Schönhage et Strassen en 1971, est basé sur des idées empruntées à l'analyse de Fourier. Plus surprenant, on ne connaît toujours pas à l'heure actuelle le nombre minimal de multiplications requises pour effectuer le produit de deux matrices de taille n. On sait seulement que ce nombre est de l'ordre de n^r, où la valeur de r s'affine peu à peu : $r = 2{,}81$ en 1969, $r = 2{,}79$ en 1978, $r = 2{,}78$ en 1979, $r = 2{,}55$ puis $r = 2{,}53$ en 1981, $r = 2{,}50$ en 1982, $r = 2{,}48$ puis 2,38 en 1987, etc. L'apparition des machines parallèles a par ailleurs changé les données du problème. Les algorithmes efficaces sur ce type de machines ne sont pas nécessairement ceux qui étaient efficaces sur des machines à un seul processeur.

L'expérience a rapidement montré aux informaticiens que malgré la puissance de leurs machines, certains problèmes restaient hors d'atteinte. Ce constat a conduit les chercheurs à donner une définition formelle de la difficulté d'un problème : c'est le point de départ de la théorie de la complexité. L'un des grands succès de cette théo-

rie est d'avoir démontré que nombre de problèmes concrets de nature très différente sont du même ordre de difficulté. Le *problème du voyageur de commerce* et le *problème du sac à dos* (étant donné un ensemble d'objets de poids donné, peut-on sélectionner un sous-ensemble dont le poids total soit exactement la charge limite autorisée du sac à dos ?) en sont des exemples. Le catalogue actuel doit certainement contenir plus de mille problèmes. Lorsque les questions sont vraiment trop difficiles pour être entièrement résolues, on peut chercher des solutions approchées. C'est l'un des objectifs de la recherche opérationnelle. Une autre solution est l'utilisation d'algorithmes dits probabilistes, qui font intervenir un ou plusieurs choix aléatoires. Ce type d'algorithmes est très utilisé en cryptographie.

La théorie des graphes et les mathématiques discrètes interviennent dans de nombreux domaines de l'informatique : en algorithmique bien sûr, mais aussi dans la modélisation des réseaux, en compilation, dans la conception des circuits, etc. Les algorithmes sur les graphes constituent d'ailleurs un chapitre important de l'algorithmique auquel se ramènent de très nombreux problèmes pratiques.

Au début, les ordinateurs manipulaient presque exclusivement des lettres et des chiffres. Aujourd'hui, ils manipulent aussi des objets en trois dimensions et l'animation en temps réel sera bientôt disponible sur micro-ordinateurs. Cette évolution a suscité la création d'une nouvelle discipline, la géométrie algorithmique. Ses applications touchent à tous les aspects géométriques de l'informatique : vision par ordinateur, réalisation de circuits à très haute intégration, robotique, etc.

Une des activités fondamentales des informaticiens est d'écrire des programmes. Pour ce faire, ils utilisent des langages dont il existe une grande variété. Plusieurs centaines de chercheurs travaillent à l'élaboration des langages du futur. La plupart de ces travaux sont fondés sur la logique mathématique. L'un des objectifs déclarés est de faciliter la "preuve" du programme, c'est-à-dire la vérification, si possible automatique, que le programme réalise bien ce que l'on attend de lui.

Le calcul formel doit être nettement distingué du calcul numérique. Il manipule des objets mathématiques et non pas des nombres. Un système de calcul formel est capable de factoriser des entiers ou des polynômes, de développer une fonction en série, de calculer sa dérivée, de résoudre des équations différentielles, bref, de réaliser le genre d'exercices que l'on propose aux étudiants en mathématiques. Mais l'utilisation du calcul formel ne s'arrête pas à ce niveau. On peut aussi manipuler des concepts mathématiques plus abstraits (groupes, idéaux), ou créer soi-même les objets sur lesquels on désire travailler.

La théorie des automates et des langages est née, dans les années cinquante, d'une tentative de formalisation du langage naturel d'une part, et des modèles neuronaux d'autre part. Elle s'est depuis développée en une discipline autonome, sous l'impulsion notamment en France de M.- P. Schützenberger, qui en a solidement établi les bases théoriques. Elle permet de modéliser des situations extrêmement diverses. Ses applications vont de la compilation à la sémantique des langages synchrones, en passant par le codage, la compression des données ou la manipulation des documents. Elle s'applique aussi bien à l'analyse des chaînes moléculaires rencontrées en biologie, qu'au traitement des chaînes de caractères.

Les informaticiens font donc grand usage des mathématiques. Ils se tournent aussi bien vers des domaines bien établis comme l'algèbre, l'analyse ou les probabilités pour l'étude des algorithmes, la géométrie pour la géométrie algorithmique, la théorie des nombres pour la cryptographie, la logique pour la sémantique des langages, que vers des disciplines plus récentes comme la théorie des graphes ou les mathématiques discrètes. Lorsque cela ne suffit pas, les informaticiens n'hésitent pas à créer une théorie sur mesure. C'est ce qui s'est produit pour la complexité et pour les automates.

Il est amusant de constater en retour l'influence de l'informatique sur les mathématiques. Les logiciens coopèrent très fréquemment avec les laboratoires d'informatique (quand ils ne deviennent pas eux-mêmes informaticiens). Des mathématiciens célèbres comme S. Eilenberg et J.- H. Conway ont chacun écrit un traité de théorie des automates et S. Smale a proposé une extension de la théorie de la complexité. Enfin, les mathématiciens attribuent désormais tous les quatre ans, outre les traditionnelles médailles Fields, un prix Nevanlinna qui récompense un mathématicien travaillant sur les fondements de l'informatique.

Jean-Eric Pin, directeur de recherche au CNRS.

L'ANALYSE D'ALGORITHMES

Combien de temps et de capacité de mémoire faut-il pour qu'un programme informatique fournisse le résultat ? Comment peut-on diminuer ce temps ? Les réponses à ces questions sont fournies par l'analyse d'algorithmes.

ANALYSIS OF ALGORITHMS - One program may be slower than another in the worst of cases and nevertheless be faster on average. Algorithmic analysis provides a means for theoretically determining computation time and storage needs as well as for speeding up programs, sometimes even by several orders of magnitude. To do so, advanced mathematical techniques are used.

Philippe Flajolet

La problématique de l'étude des algorithmes a été entièrement renouvelée avec l'arrivée des ordinateurs. L'étalon des calculs faisables est passé du millier d'opérations (dans les années 50), au million (dans les années 70), puis au milliard (dans les années 80), ce qui a permis d'envisager des classes de méthodes entièrement nouvelles soit par le volume, soit surtout par la nature des données traitées. Se sont ainsi dégagées successivement les possibilités de calculer sur des données non numériques élémentaires (problèmes de tri ou de recherche), puis textuelles, géométriques (données multidimensionnelles), voire symboliques (calcul formel, démonstration automatique par exemple).

La mise au point d'algorithmes efficaces, 10 à 100 fois plus rapides que l'algorithme de base, a provoqué un foisonnement de travaux, particulièrement sur les arbres, sans doute la notion mathématique la plus importante pour l'informatique, et dont il existe toute une variété : arbres de recherche (pour la recherche en table), arbres paginés (méthodes d'accès rapide sur disque), arbres préfixes ou suffixes (traitement de données textuelles ou génétiques), arbres de syntaxe (compilation), arbres de quadrants (gestion de données géométriques) et même des protocoles en arbres pour la communication dans les réseaux informatiques.

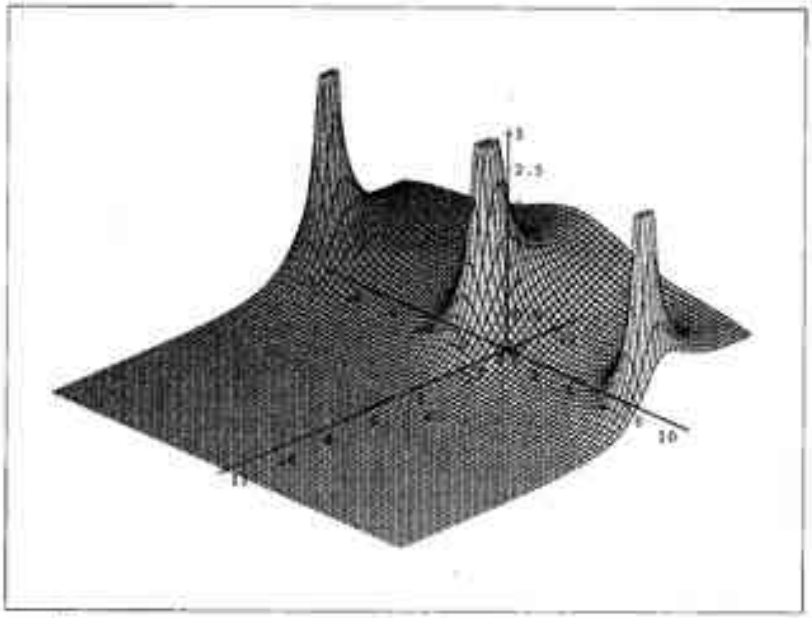
A titre d'exemple, l'utilisation de telles structures de données finement optimisées a permis à une équipe de Waterloo (Gonnet) de réaliser l'informatisation complète du grand *Oxford English Dictionary* sur station de travail ; les temps de réponse sont de quelques secondes seulement, même pour des requêtes de forme particulièrement complexe. Le corpus est pourtant gigantesque : plusieurs dizaines d'hommes-année en simple volume de saisie ! Par exemple, un lexicographe a pu déterminer l'éclosion du concept de maladie psychosomatique au début du XX^ème^ siècle en quelques heures d'interaction seulement avec le système. La recherche en algorithmique utilise une grande variété de méthodes parmi lesquelles les décompositions récursives, les méthodes à adressage direct, les filtres, les méthodes probabilistes.

Temps et mémoire de calcul

L'analyse d'un algorithme consiste principalement à prédire son comportement, en temps ou en mémoire de calcul, sur l'ensemble de tous les cas possibles (analyse dans le pire cas, analyse en moyenne) ou encore sur des ensembles de données obéissant à certaines lois de hasard. A un premier niveau d'approximation, on se limite au cas le pire, fondé sur la recherche de configurations extrêmes qui forcent un mauvais comportement de l'algorithme, ainsi qu'à de simples estimations d'ordres de grandeur. On établit alors que le "tri rapide" (*Quicksort*) et le "tri fusion" (voir figure) présentent, dans le pire des cas, des temps de calcul respectivement proportionnels à n^2 et $n \log n$, n étant le nombre d'objets à trier.

C'est pourtant le tri rapide qui a été choisi par les créateurs du système Unix, malgré cette caractéristique a priori défavorable (pour n grand, n^2 croît beaucoup plus vite que $n \log n$). C'est qu'à la suite des travaux de Hoare, Knuth et Sedgewick, il a été montré que sur un cas moyen, cet algorithme a un "coût" proportionnel à $n \log n$ (avec de plus un tout petit coefficient de proportionnalité). Le tri rapide est donc meilleur en moyenne que le tri fusion, bien qu'il soit moins bon dans le pire des cas...

Depuis les années 1960 sont apparus en analyse combinatoire des cadres mathématiques qui permettent de relier directement structures combinatoires et fonctions génératrices de dénombrement associées. Il découle des travaux de Rota aux Etats-Unis et de l'Ecole française (Foata et Schützenberger notamment) que les constructions combinatoires les plus utiles à l'informatique (les unions ou produits de types, la formation de listes



Le comportement oscillant de l'algorithme de tri fusion se lit sur les singularités régulièrement espacées d'une certaine série génératrice des coûts (trois singularités sont représentées ici).

et ensembles, la définition de structures récursives) possèdent des traductions en opérateurs sur les fonctions génératrices.

Il est ainsi possible d'accéder au coût d'un algorithme par le biais d'opérations entre séries génératrices. Dans les cas favorables, on peut même développer des "calculs de complexité" qui conduisent à des prédictions exactes du coût moyen d'un algorithme. Dans les cas les plus favorables, les équations se résolvent même en termes de fonctions classiques de l'analyse mathématique qui permettent de déterminer, avec des formules exactes, les coûts moyens de certains algorithmes.

Par la variété des objets combinatoires qu'elle manipule, l'analyse d'algorithmes a entraîné un renouveau important de l'analyse combinatoire. Des domaines nouveaux se sont ouverts comme l'étude des arbres aléatoires (qui entretient des liens étroits avec les processus de branchement probabilistes) ou celle des graphes aléatoires, domaine fondé par Erdös et Renyi vers 1959, et dont le développement spectaculaire répond à la fois à une problématique mathématique interne et à la nécessité de comprendre les phénomènes probabilistes issus de l'optimisation combinatoire.

Passer du comportement global à celui au voisinage des singularités

Dans le cas le plus général, la complexité des équations de séries génératrices de dénombrement et de coût ne permet pas de résolution directe. En s'inspirant d'une démarche dégagée clairement en théorie analytique des nombres, on peut réduire ces équations, par nature globales, en se limitant à leur comportement local au voisinage de points spéciaux, singularités ou points critiques.

> En France, les points discutés dans cet article sont notamment développés à l'intérieur du PRC/GDR *Mathématiques et informatique*. Celui-ci regroupe une quarantaine d'équipes appartenant aux deux disciplines, les principaux thèmes de recherche étant : algorithmique, automates, calcul formel, combinatoire et logique.

On fait alors appel à des méthodes issues de l'analyse complexe (analyse de singularité, transformations intégrales), et à diverses méthodes d'étude d'équations fonctionnelles ou différentielles, déjà bien connues en mathématiques (itération, fonctions holonomes, équations aux différences) ou entièrement nouvelles.

Un très grand nombre des algorithmes de base, d'utilisation générale en informatique, sont désormais bien compris quant à leurs performances de calcul. L'analyse a souvent permis des optimisations extrêmement fines, avec des gains de performances d'un à plusieurs ordres de grandeur. Par exemple, les méthodes de hachage extensible ou dynamique autorisent des accès quasi-directs à de très gros volumes de données. Presque tout texte en langue naturelle peut de manière routinière (et rapide) être comprimé dans des rapports de 2 ou 3, et de gros corpus de données textuelles ou de grosses bases de données peuvent désormais être soumis à des interrogations complexes.

L'analyse d'algorithmes emprunte beaucoup aux mathématiques. Elle n'a pas cependant un simple rôle de client. Elle renouvelle aussi la problématique de nombreux chapitres de cette discipline, quand elle ne lui apporte pas une aide. Le calcul du nombre π, l'arithmétique calculatoire ou la localisation de zéros de polynômes bénéficient de la transformée de Fourier rapide, développée à l'origine par les informaticiens pour le traitement du signal. Les progrès sur les tests de primalité et la factorisation d'entiers ont permis de concevoir des systèmes cryptographiques à clés publiques, possédant de remarquables propriétés de sécurité. Le calcul formel, lui-même appelé à un grand développement, se situe à la charnière entre l'algorithmique (structures de données), la logique et la sémantique (simplification, réécriture, mécanismes d'évaluation dans les langages de programmation) et l'algèbre ou l'analyse (équations polynomiales, intégration formelle, équations différentielles).

On assiste ainsi, en algorithmique en général, et en particulier en analyse d'algorithmes, à un bouillonnement d'idées qui témoigne de la vivacité scientifique du domaine et qui met en relief de très nombreuses situations d'applications inattendues pour des mathématiques traditionnellement considérées comme pures.

Philippe Flajolet, directeur de recherche à l'INRIA, directeur du PRC Mathématiques et Informatique (GDR 967 CNRS), BP 105, 78153 Le Chesnay Cedex.

LES AUTOMATES FINIS

Le concept d'automates finis a un double intérêt. Il intervient en pratique dans la construction de nombreux algorithmes, notamment dans les traitements de texte. Mais il est aussi l'objet d'intenses investigations théoriques, car il est à la base de la modélisation mathématique de nombreux concepts issus de l'informatique.

Maxime Crochemore
Dominique Perrin
Jean-Eric Pin

> *FINITE-STATE AUTOMATA - Finite-state automata are used for many software applications, in particular word processing. In some cases, dictionaries for instance, they enable significant compression of data (a ratio of nearly 1 to a hundred) without any loss of information. The theory of automata has developed significantly in recent years thereby considerably broadening its scope of application. For instance, certain critical parts of programs which must be operational at all times, such as operating systems, can be formalized.*

Les automates finis constituent un modèle mathématique très utilisé en informatique : on les retrouve dans nombre d'algorithmes (notamment ceux d'un traitement de texte). Ils interviennent dans les processus de compilation, dans l'étude des "schémas de programme", dans la modélisation des circuits booléens et du temps réel. Ils jouent également un rôle important dans la formalisation de certains systèmes non linéaires en automatique. Ils constituent en outre le modèle de machine le plus élémentaire et le plus concret, sur lequel sont basés la plupart des modèles de machines plus puissants (automates à pile, avec sortie, à bandes, etc., jusqu'aux célèbres machines de Turing). Leur étude a été entreprise vers la fin des années cinquante et n'a cessé depuis de se développer. Les

motivations des premiers travaux se rattachaient à la cybernétique, alors florissante : il s'agissait de donner un modèle du cerveau fondé sur des "réseaux de neurones". Mais, dès 1954, Kleene avait pressenti l'importance des automates finis : "Savoir ce qu'un automate fini peut faire et ce qu'il ne peut pas faire est une question qui possède un intérêt mathématique intrinsèque, mais qui peut aussi contribuer à une meilleure compréhension de problèmes purement pratiques". L'article où figure cette prédiction est désormais considéré comme le point de départ véritable de la théorie des automates, car il contenait également un remarquable résultat qui délimite de façon très précise la puissance des automates finis ("ce qu'un automate fini peut faire et ce qu'il ne peut pas faire…").

Pour avoir une idée intuitive de ce qu'est un automate, on peut penser à un ascenseur rudimentaire, muni seulement de deux boutons : le bouton M permet de monter d'un étage (et n'a pas d'action visible lorsque l'ascenseur est au dernier étage), et le bouton D permet de descendre d'un étage (et n'a pas d'action visible au rez-de-chaussée). Si l'immeuble a deux étages, on peut représenter cette situation sur la figure suivante :

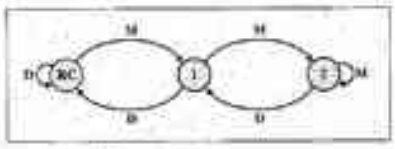


Fig. 1 - Un automate.

On vient de donner un exemple d'automate dans lequel les symboles RC, 1 et 2 sont les états, états sur lesquels agissent les lettres M et D.

Les automates sont souvent utilisés pour représenter commodément certains ensembles de suites de symboles ; les automates de la figure 2 représentent tous les deux l'ensemble des suites de bits dans lesquelles deux bits consécutifs ne peuvent être égaux à 1 (une suite de symboles ne peut être "lue" sur un automate qu'en suivant un chemin formé de flèches consécutives).

Il existe cependant une grande différence entre l'automate du haut et celui du bas. Sur l'automate du bas, l'action d'un symbole (0 ou 1) sur un état (A ou B) est parfaitement déterminée. Par contre, l'automate du haut offre un choix : dans l'état A, la lecture du symbole 0 permet soit de rester dans l'état A, soit de passer dans l'état B. On dit que l'automate du bas est déterministe et que celui du haut est non déterministe. Cette distinction est d'une grande importance théorique et pratique. La théorie montre en effet que tout automate non déterministe peut être simulé par un automate déterministe. Mais cette simulation peut se révéler coûteuse, puisqu'il faut passer de n états à 2^{n-1} états dans le cas le plus défavorable.

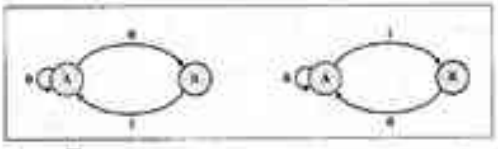


Fig. 2 - Deux automates pour les suites de bits ne contenant pas deux 1 consécutifs.

Traiter des chaînes de caractères

Une application courante des automates finis est le traitement des chaînes de caractères dans un texte. C'est un domaine où des progrès importants ont été réalisés ces dernières années, tant au niveau des logiciels que des algorithmes. Le champ d'application est très vaste puisque, fait remarquable, des méthodes similaires s'appliquent à la fois aux problèmes du traitement de texte au sens usuel et à celui des données génétiques de la biologie moléculaire.

La localisation d'une chaîne de caractères à l'intérieur d'une autre chaîne plus longue est un problème de base en informatique. Sa résolution a suscité un grand nombre de travaux théoriques liés d'une part à l'étude combinatoire des suites de symboles, et d'autre part à la description et à l'analyse d'algorithmes. Le point de départ de ces travaux est le fait que l'ensemble des chaînes de caractères contenant une chaîne donnée peut être représenté par un automate fini. Les résultats les plus diffusés dans le public sont ceux qui ont contribué en partie au succès du système d'exploitation UNIX, mais un grand nombre d'autres réalisations ont vu le jour sur des sujets voisins comme la compression de texte, le codage ou la recherche documentaire.

Les problèmes posés par l'analyse des chaînes moléculaires rencontrées en biologie sont souvent les mêmes que ceux rencontrés dans le traitement des chaînes de caractères. Ainsi, l'algorithme de recherche de la plus longue sous-suite commune à deux suites données est nécessaire à la fois à la comparaison des séquences moléculaires et à celle des différentes versions d'un fichier. En effet, la présence de longues sous-suites communes est utilisée dans les deux cas comme critère de ressemblance. Un Groupement de Recherche "Informatique et Génome" est en cours de création pour favoriser la collaboration sur ce type de problèmes.

La compression des données et les dictionnaires électroniques

Une autre application importante des automates est la représentation d'ensembles en machine. Prenons par exemple l'ensemble suivant, extrait d'un dictionnaire de mots croisés :

{BAL, BALS, BAN, BANS, PAL, PALS, PAN, PANS, DO, DON, DONS, DOUX}

Pour représenter cet ensemble en machine, on peut utiliser une liste, ou encore une structure d'arbre (Fig. 3).

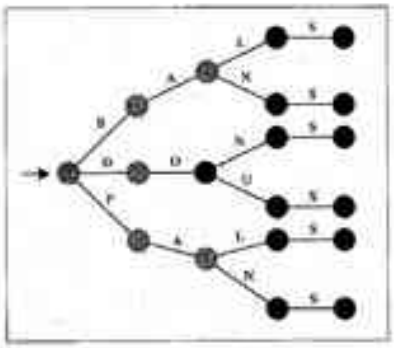


Fig. 3 - Représentation par arbre.

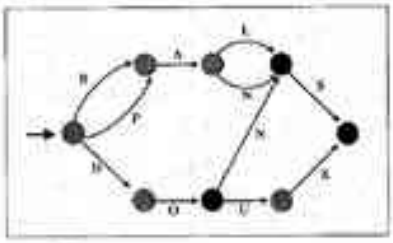


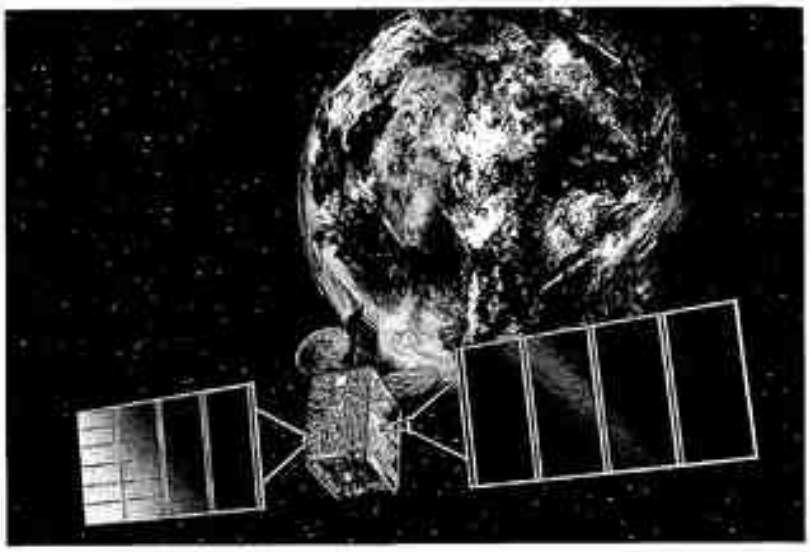
Fig. 4 - Représentation par automate.

Mais l'utilisation d'un automate (Fig. 4) aboutit à une représentation beaucoup plus compacte : dans le cas d'un dictionnaire, le gain est de l'ordre de 80 %.

Cette observation a servi de point de départ à la mise au point d'un système de consultation et de mise à jour de dictionnaires électroniques réalisés au sein des laboratoires CNRS de l'Institut Blaise Pascal à Paris. Les dictionnaires sont issus du travail effectué au Laboratoire d'automatique documentaire et de linguistique (LADL). Il s'agit d'un ensemble comprenant des listes de plusieurs unités linguistiques, allant des mots simples aux expressions composées. Les dictionnaires de mots simples comprennent de 80 000 à 500 000 formes, suivant que les mots sont représentés sous leur forme de base ou sous une forme fléchie (genre, nombre ou personne et temps pour les verbes). Les dictionnaires de ▶

mots composés sont en cours de constitution et leur taille est supérieure d'un ordre de grandeur à celle des précédents. Les systèmes d'interrogation permettent notamment de localiser dans des textes des expressions d'un type donné et de constituer un index technique des documents. Les méthodes utilisées permettent la représentation de 500 000 mots en accès direct sur 300 Koctets de mémoire. Elles reposent sur une méthode de compression des textes adaptée aux lexiques qui donne un taux de compression impressionnant : un bit pour deux caractères.

De façon plus générale, la compression de données est un problème d'une grande importance économique, dont l'utilisation la plus connue du grand public est la télécopie, et qui a connu des résultats importants dans les dernières années. A titre d'exemple, une collaboration entre le CNRS et le CNES a permis de comprimer les données de gestion du satellite TDF1 avec un gain supérieur à 90 % (rapport de compression de 1 à 13). Il s'agit, contrairement à la plupart des méthodes utilisées en compression d'images ou du son, d'une compression sans perte d'information. La compression des données s'applique également au stockage et à l'archivage des informations, car l'augmentation des capacités d'archivage va de pair avec la croissance des besoins des utilisateurs.



Une collaboration entre le CNRS et le CNES a permis de comprimer les données de gestion du satellite TDF1 avec un rapport de compression de 1 à 13. (CNES - Illustration D. Ducros).

Une utilisation courante en compilation et manipulation de fichiers

Les automates finis sont largement utilisés en compilation. Un compilateur est un programme qui "traduit" un programme écrit dans un langage (par exemple PASCAL, FORTRAN, C) en un autre langage (par exemple en langage machine). Les automates finis apparaissent surtout dans la première phase de la compilation qui consiste à repérer les mots-clés du langage (*begin, end, for*, etc.) et à effectuer diverses tâches auxiliaires : élimination des blancs et des commentaires, numérotation des lignes, messages d'erreur, etc. Par exemple, dans le langage de programmation C, le caractère /* (respectivement */) indique un début (une fin) de commentaire. Les commentaires en C sont complètement décrits par l'automate suivant, dans lequel le point désigne tout caractère valide autre que / ou * (Fig. 5).

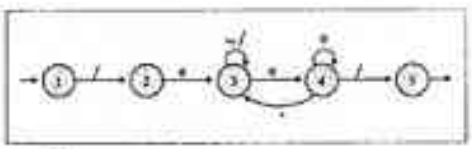


Fig. 5 - Reconnaissance des commentaires en C.

Les automates finis interviennent également dans les systèmes de recherche d'information et dans les langages d'interrogation de données ou de manipulation de fichiers. Ils ont été également utilisés pour décrire certaines imperfections dans les circuits imprimés.

Enfin, sous des formes diverses, les automates jouent un rôle très important dans la modélisation des concepts de l'informatique. Ils interviennent notamment dans l'analyse des processus séquentiels et dans la formalisation du parallélisme. Certains concepts sont issus de travaux de nature extrêmement théoriques, mais ont une grande portée pratique : par exemple, les automates travaillant sur des arbres sont à la base du langage PROLOG III.

Un concept riche de conséquences théoriques

La définition élémentaire des automates – tout comme celle des groupes en mathématiques – cache une très grande complexité et une très grande richesse. De ce fait, la théorie des automates est un domaine dans lequel sont poursuivies de nombreuses recherches théoriques. Des résultats spectaculaires ont été obtenus ces dernières années, ils résolvent pour certains d'entre eux des problèmes ouverts depuis des dizaines d'années.

L'un des thèmes de recherche consiste à raffiner le résultat originel de Kleene. En imposant des conditions sur les automates, on définit de nouvelles sous-classes, dont il s'agit de déterminer le pouvoir expressif. Le premier résultat de ce type a été obtenu en 1965 par Schützenberger qui a caractérisé les automates "sans compteur", et de nombreuses autres classes ont été caractérisées depuis lors.

Un autre domaine dans lequel des recherches sont poursuivies actuellement est celui des systèmes logiques adaptés aux automates. Les recherches portent sur les algorithmes de transformation de formules logiques en automates et inversement. Elles s'appliquent notamment à la fabrication de systèmes de vérifications de formules de logique portant sur des systèmes ayant un nombre fini d'états. Par exemple, il est très facile de convertir une description telle que "l'ensemble des suites de bits contenant une séquence de cinq 1 consécutifs mais aucune séquence de trois 0 consécutifs" en une formule de logique. On sait alors convertir cette description logique en un automate fini et vice-versa. Les recherches portent notamment sur les complexités comparées de ces descriptions. On sait par exemple que les automates sans compteur évoqués plus haut correspondent très exactement aux formules du "premier ordre" de la logique.

Nous avons implicitement supposé jusqu'ici que les automates s'appliquaient aux suites finies de symboles, mais une part importante de la théorie concerne également les suites infinies. Le résultat fondamental de ce domaine est un théorème dû à R. McNaughton datant des années soixante. Il montre que les automates finis reconnaissant des suites infinies peuvent encore, comme dans le cas

des suites finies, être simulés par des automates déterministes. Récemment, un jeune chercheur israélien a obtenu une nouvelle construction qui améliore la complexité de l'algorithme issu de la construction originale de McNaughton. De nombreux chercheurs en France ont apporté des contributions décisives dans ce domaine, notamment à Paris (LITP et CNET) et à Bordeaux (LABRI).

Ce type de recherche permet de séparer le champ du possible de celui de l'impossible et permet surtout de dégager des concepts qui sont utilisés par ailleurs. Enfin, les automates jouent un rôle très important dans la modélisation des systèmes à événements discrets. De nombreux travaux portent actuellement sur l'utilisation de ces modèles dans l'étude du parallélisme, des circuits booléens ou de la vérification formelle de programmes.

Maxime Crochemore, professeur à l'Université de Marne la Vallée, Laboratoire d'informatique théorique et programmation (URA 248 CNRS), Institut Gaspard Monge, 2, allée Jean Renoir, 93160 Noisy le Grand.

Dominique Perrin, professeur à l'Université Paris VII et à l'ESIEE, directeur du Laboratoire d'informatique théorique et programmation (URA 248 CNRS) et directeur scientifique de l'ESIEE, Cité Descartes, BP 99, 93162 Noisy le Grand Cedex.

Jean-Eric Pin, directeur de recherche au CNRS, Laboratoire d'informatique théorique et programmation (URA 248 CNRS), mis en disponibilité auprès du Centre de recherches et développement de BULL, rue Jean-Jaurès, 78340 Les Clayes-sous-Bois.

COMPLEXITÉ ET CRYPTOGRAPHIE

Dans la plupart des cas, un programme trop long à exécuter est inutilisable. Mais cette propriété devient un avantage quand on veut chiffrer un message qui ne peut être décrypté que par son destinataire, parce qu'il est le seul à posséder les données qui rendent efficace le programme de déchiffrement.

COMPLEXITY AND CRYPTOGRAPHY - The time required to run a program depends on the quantity of data n to be processed. Programs in which computing time is increased by an n polynomial are generally executable. This is not the case when time varies as an exponent of n, leading to a "combinatorial explosion". With some types of data it is possible to pass from one type to the other. In this way, "public-key" cryptography can be used whereby message sending devices transmit ciphered documents that can be red only by the addressee.

Jacques Stern

Le mot complexité recouvre en informatique une notion intuitive assez claire : un problème est d'autant plus complexe que la suite des calculs qui mène au résultat est longue. Pour quitter le qualitatif, les théoriciens ont imaginé de considérer des machines abstraites, pour lesquelles le temps de calcul est défini rigoureusement, comme la machine de Turing.

Le pas décisif de la modélisation consiste à évaluer un algorithme par le comportement asymptotique de la fonction qui mesure le temps de calcul. A cet égard, deux fonctions f et g telles que $\lim_{n\to\infty} f(n)/g(n) = 1$ fournissent la même complexité. Ce point de vue peut paraître étrange, puisqu'on mesure des phénomènes finis par essence, en examinant ce qui se passe à l'infini. De fait, les justifications en sont largement expérimentales et liées au fait qu'en général n est assez grand pour que l'analyse soit pertinente.

Pour beaucoup de problèmes courants (problèmes de tri, plus court chemin...), un algorithme en $O(n^k)$, c'est-à-dire dont le temps de calcul est majoré par un polynôme en n (nombre d'éléments), est connu. On est ainsi conduit à identifier algorithmes praticables et temps polynomial.

Ce faisant, on élimine certains problèmes pour lesquels des questions en apparence très simples conduisent à des solutions dont la mise en œuvre requiert un temps gigantesque. Même en ayant recours à des ordinateurs très puissants, même en en faisant coopérer un grand nombre et même compte-tenu des progrès attendus, il est exclu qu'on vienne à bout des problèmes conduisant à une "explosion combinatoire", c'est-à-dire nécessitant 2^n pas de calcul.

Parmi les problèmes qui, en l'état actuel de nos connaissances, conduisent à l'explosion, nombreux sont ceux qui sont formulés comme la recherche d'une solution à un problème combinatoire "simple". Seule l'énumération des solutions est coûteuse : la vérification d'une solution correcte est, quant à elle, polynomiale. Les problèmes de cette forme constituent la classe NP, ces deux lettres indiquant que si l'on trouve la solution de façon Non-déterministe (c'est-à-dire en commençant miraculeusement l'énumération par une solution correcte), on est ramené au temps Polynomial.

Un exemple est le "problème des circuits logiques" : étant donné un circuit construit à l'aide d'un grand nombre de portes (ET, OU, NON...) et comportant plusieurs entrées et une sortie, existe-t-il au moins une distribution 0/1 sur les entrées pour laquelle la sortie est à 1 ?

Le célèbre problème P = NP ? est la question de savoir si l'on peut exclure l'existence d'algorithmes polynomiaux pour certains problèmes de la classe NP. Ce problème posé depuis une vingtaine d'années reste ouvert, mais il est certain qu'il a énormément stimulé la recherche dans le domaine. On se bornera ici à évoquer le travail de Razborov, qui a valu à son auteur l'attribution du prix Nevanlinna. Ce travail s'inscrit dans une problématique connue comme l'approche non uniforme de la question P = NP : si l'on se restreint aux données de taille n, le résultat du calcul d'une machine de Turing de complexité polynomiale est obtenu par un circuit logique dont le nombre de portes est polynomialement borné en fonction de n. Pour démontrer la conjecture P ≠ NP, il suffirait donc d'exhiber un problème de la classe NP pour lequel toute famille de circuits C_n donnant la réponse correcte pour les données de taille n soit de taille exponentielle en n. On n'en est pas là : Razborov a fait une percée remarquable en traitant le seul cas des circuits monotones (construits avec les portes ET et OU).

La complexité gardienne du secret

Qu'un problème exige trop de temps de calcul pour être pratiquement résolu peut devenir un avantage. Un message qui ne peut être décrypté que par un programme trop long à mettre en œuvre restera secret pour qui n'en détient pas la clé. ▶

La cryptographie ou science des messages secrets est confrontée aujourd'hui à une formidable demande en provenance de divers horizons : assurances, banques, télécommunications, etc.

Appelons fonction à sens unique une fonction calculable par un algorithme polynomial, mais impossible à inverser par un algorithme polynomial, même probabiliste (où certains tests d'un programme sont faits au hasard), et même en se fixant l'objectif limité d'inverser seulement une partie significative des cas possibles.

Mentionnons qu'on ne sait pas prouver l'existence d'une telle fonction, laquelle donnerait une réponse au problème P = NP ? On ne dispose donc, en l'état actuel de nos connaissances, que de candidats au rôle. Parmi eux, le produit *mn* de deux entiers ayant une centaine de chiffres décimaux (factoriser un nombre est difficile, le record à ce jour enregistré est en deçà de 150 chiffres décimaux !).

La notion de fonction à sens unique a de multiples applications et elle permet de résoudre des problèmes amusants : comment jouer à pile ou face par téléphone (sans tricher) ?... ou sérieux : comment valider par une signature digitale des transferts de fonds électroniques. Moyennant la mise en place d'une "trappe", c'est-à-dire d'une information secrète disponible à certains utilisateurs seulement, elle autorise une méthode cryptographique dite à clé publique. Le chiffrement peut être effectué par quiconque, et seul le déchiffrement nécessite la connaissance d'une donnée secrète. Le système de codage RSA de Rivest, Shamir et Adleman (1978) réalise un tel code. Il est fondé sur l'utilisation d'un produit n de deux nombres premiers. L'entier n est public, mais sa factorisation ne l'est pas.

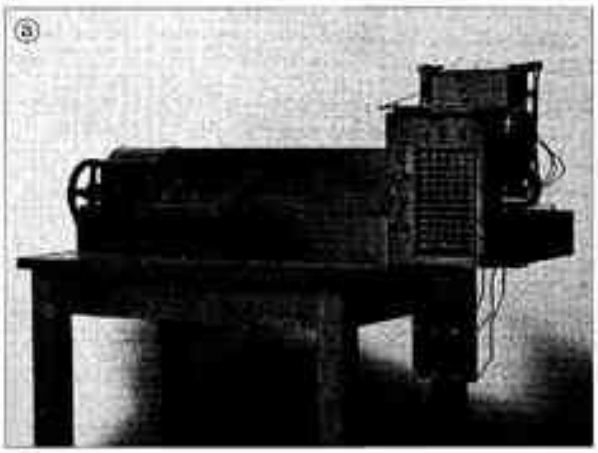
La notion de fonction à sens unique permet également d'imaginer une interaction sous forme de questions/réponses entre deux acteurs cryptographiques aboutissant à l'identification de l'un par l'autre. Un secret est en jeu, mais, à l'issue du protocole, aucune information sur ce secret n'a été divulguée. On parle de *zero-knowledge*.

Motivée par la cryptographie, la notion d'interaction a récemment produit de spectaculaires résultats théoriques comme celui de Shamir (1990), connu sous le nom de IP = PSPACE, qui identifie les problèmes relevant d'une interaction polynomiale comme ceux dont la vérification exhaustive requiert un espace mémoire polynomial (et donc vraisemblablement un temps exponentiel).

Ce résultat s'appuie sur une technique dite d'arithmétisation par les polynômes de bas degré, qui introduit entre les diverses parties d'une longue "preuve" une forte cohérence. Des améliorations toutes récentes montrent que la vérification d'une telle preuve se réduit essentiellement à tester quelques bits d'information çà et là.

Les recherches que l'on vient d'évoquer sont en plein développement. Les notions sont riches d'applications pratiques (par exemple dans les transactions avec cartes à mémoire) mais, plus important peut-être, elles ouvrent sur le concept même de preuve un débat qui dépasse le cadre de l'informatique théorique : d'une certaine façon, elles réhabilitent la notion d'induction (ou inférence statistique) face à la notion mathématique de déduction.

Jacques Stern, directeur de recherche au CNRS, Groupe de recherche en complexité et cryptographie, Laboratoire d'informatique de l'École normale supérieure (URA 1327 CNRS), 45, rue d'Ulm, 75230 Paris Cedex 05.



Ces trois machines, a) la machine à statistiques de Frederik R. Bull (1923), b) une machine BULL des années cinquante et c) un DPS7000 (1992), illustrent la formidable progression de la puissance des ordinateurs. Toutefois, malgré ces progrès spectaculaires, certains problèmes dits "exponentiels" restent hors de portée et le resteront dans le futur (documents Bull).

SÉMANTIQUE ET PREUVE EN PROGRAMMATION

La sémantique des langages de programmation a pour objectif de comprendre le sens de chacune des parties qui composent un programme. Elle veut fournir à ses utilisateurs les moyens de raisonner sur les spécifications des programmes et de démontrer l'adéquation entre un programme et ses spécifications.

SEMANTICS AND PROOF IN PROGRAMMING - Lambda-calculus invented by the mathematician Church around 1930 was used by Landin in the 60's to translate the programming language Algol60 into mathematical notation. This was the first semantics formalism. Since then, Logics has remained the main source of inspiration for providing semantics formalisms. The various semantic approaches have two major purposes: producing tools automatically and providing proof techniques.

Guy Cousineau

La sémantique dénotationnelle d'un langage de programmation associe à tout programme la fonction qu'il calcule (son sens). La première sémantique a été inventée en 1965 pour Algol60 par John Landin. Elle était obtenue par traduction d'Algol60 dans une notation mathématique, le λ-calcul. Ce travail de pionnier a réactivé les études sur le λ-calcul et a débouché sur le développement d'une technique de définition sémantique : la sémantique dénotationnelle de Scott et Strachey. Algol60 introduisait des concepts nouveaux comme la structure de bloc, les paramètres par valeur et par nom et la récursivité dont la compilation posait des problèmes, ce qui conduisait naturellement à des interrogations de nature sémantique.

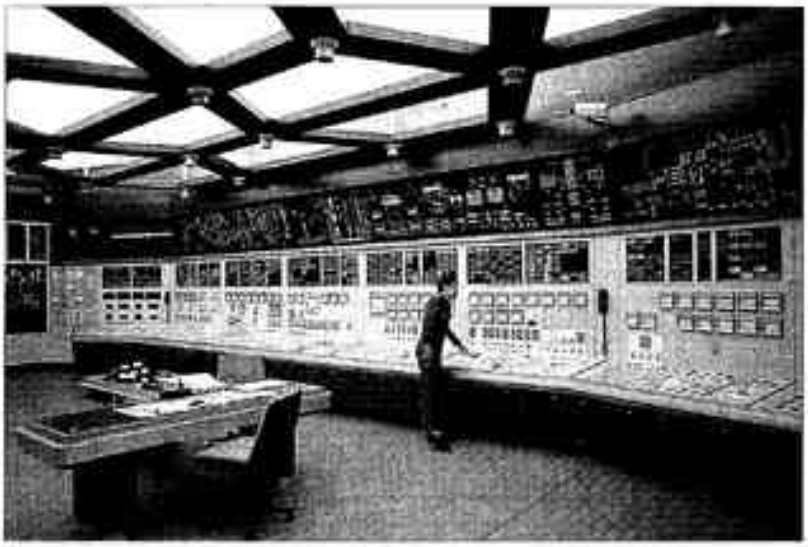
A long terme, le but de la recherche en programmation est de fournir des formalismes, des méthodes et des outils pour développer des programmes dont la correction vis-à-vis de spécifications formelles peut être démontrée. Les définitions sémantiques visent à fournir le moyen de raisonner sur les spécifications et les programmes, et d'en démontrer des propriétés. Etant donné la taille des spécifications et des programmes, les preuves de correction ne pourront pas être faites à la main. Il faut donc se placer dans la perspective de preuves en grande partie automatisées, et en tout cas fortement assistées.

Des langages à la sémantique définie simplement

Les langages couramment utilisés n'ont pas été conçus en fonction des méthodes de preuves et sont donc mal adaptés à leur développement. Une des retombées importantes de la recherche sur la sémantique et les méthodes de preuve est donc de proposer des nouveaux langages dont la sémantique est définie simplement et qui sont bien adaptés aux preuves. C'est le cas par exemple des langages fonctionnels ou des langages logiques.

Par ailleurs, la recherche de l'effectivité dans les définitions sémantiques conduit à des formalismes qui peuvent être directement utilisés pour produire des outils. Par exemple, la traduction de Landin d'Algol dans le λ-calcul, noyau des langages fonctionnels, est directement exécutable. Cette traduction fournit donc un moyen d'exécuter Algol. Plus généralement, on peut s'appuyer sur des définitions sémantiques pour fabriquer automatiquement des outils comme des interpréteurs ou des compilateurs, pour un langage de programmation donné. Ces outils sont corrects par construction puisqu'ils s'appuient sur la définition sémantique du langage. Il est donc devenu possible, à partir de la définition formelle d'un langage de programmation (syntaxe+sémantique), de prototyper automatiquement les utilitaires permettant de se servir du langage.

En pratique toutefois, ce type de sémantique dénotationnelle est assez incommode. Aussi s'oriente-t-on vers des formes sémantiques plus opérationnelles, permettant de préciser comment la fonction associée au programme est effectivement calculée. Un des objectifs de la recherche est d'étudier différents niveaux de sémantiques, consistants entre eux, et permettant de raisonner sur les programmes à différents niveaux d'abstraction.



Salle de commande de la centrale nucléaire de Chinon. (Sodel, photothèque EDF, cliché J.-F. Le Coguiec). L'omniprésence de l'informatique dans le développement technologique rend chaque jour plus nécessaire le développement d'outils de programmation fiables. Les recherches en sémantique trouvent là leur principale motivation.

Une autre retombée intéressante est l'analyse automatique de programmes. Même s'il n'est pas possible aujourd'hui de démontrer complètement l'adéquation d'un programme à sa spécification, il est possible d'en démontrer automatiquement certaines propriétés partielles, qui permettent d'améliorer sa compilation ou d'en paralléliser partiellement l'exécution. Pour parvenir à ce but, on définit des sémantiques partielles qui ne gardent de la fonction calculée que les propriétés que l'on souhaite étudier.

Sémantiques totales et sémantiques partielles

Les outils de définition sémantique des langages de programmation sont fournis par les mathématiques. Il peut s'agir d'outils préexistants ou d'outils développés par les informaticiens théoriciens. Nous distinguerons les formalismes utilisés pour définir des sémantiques totales, c'est-à-dire décrivant de façon complète la fonction calculée, et les formalismes utilisés pour définir des sémantiques partielles, utilisées lorsque l'on s'intéresse à des propriétés particulières.

Dans la première catégorie, on trouve le λ-calcul, formalisme introduit par un logicien, A. Church, vers 1930, et permettant de noter de façon effective les fonctions calculables. Les termes du λ-calcul sont construits sur deux notions : la notion fonctionnelle et la notion d'application.

Dans la seconde catégorie, on trouve des formalismes très divers adaptés chacun à l'étude de propriétés spécifiques des programmes. La notion de schéma de programme permet d'isoler certaines structures des langages de programmation et d'étudier des propriétés abstraites de programmes, indépendemment des structures de données sur lesquelles ils opèrent. On fait appel pour cela à la théorie des langages et automates d'arbres. La théorie des langages et automates sur les mots est également utilisée pour étudier, par exemple, la sémantique du parallélisme en particulier celle des langages synchrones. L'algèbre des treillis intervient dans la définition de sémantiques partielles utilisées en analyse automatique de programmes.

La programmation comme branche appliquée de la logique

Les liens entre sémantique des programmes et logique mathématique sont si nombreux qu'il est permis de considérer la programmation comme une branche appliquée de la logique mathématique. La notion de fonction calculable et celle de décidabilité, introduites par les logiciens dans les années 30, définissent en quelque sorte la limite du domaine de l'informatique, domaine raffiné par la suite par les études sur la complexité. Toutefois, les liens les plus étroits entre informatique et logique se trouvent dans le domaine de la théorie de la démonstration, initiée par Gödel et Gentzen et développée de façon remarquable dans les quinze dernières années, en particulier en France par Girard. Le formalisme commun aux deux disciplines est le λ-calcul qui sert en informatique à noter des programmes, et en théorie de la démonstration à noter des preuves. De la preuve (constructive) d'une propriété du type "pour tout x, il existe y tel que $P(x, y)$", il est possible de passer à un programme calculant y à partir de x de façon à satisfaire la spécification P, tout en restant dans le cadre du λ-calcul.

Même si cette approche de la programmation est encore largement du domaine de la recherche, elle constitue un embryon de réponse au problème de la preuve, fondamental pour la programmation, et montre clairement comment inscrire ce problème dans le cadre de la théorie de la démonstration. Les systèmes de preuve développés en théorie de la démonstration (déduction naturelle, calcul des séquents) fournissent dès maintenant des formalismes couramment utilisés pour décrire des sémantiques opérationnelles dans le cadre d'environnements de programmation. La logique est donc clairement pour les chercheurs en programmation une source fondamentale d'inspiration.

Guy Cousineau, professeur à l'ENS, directeur du Laboratoire d'informatique de l'École normale supérieure (URA 1327 CNRS), 45, rue d'Ulm, 75230 Paris Cedex 05.

LA THÉORIE DES GRAPHES

Ces dernières années, la théorie des graphes est devenue indispensable en informatique, aussi bien pour modéliser un programme qu'un algorithme ou l'architecture d'un ordinateur. D'autres domaines de l'informatique ou d'autres disciplines font aussi appel à cette théorie.

Jean-Claude Bermond
Emmanuel Lazard
Dominique Sotteau
Michel Syska

GRAPH THEORY – Interconnections networks between processors in parallel machines are given us an illustration of the significance of the theory of graphs in computer science. The networks used can be classical: complete networks, ring, hypercube, grid networks. But slightly more complex networks such as De Bruijn networks present substantial advantages and may take the fore in the future.

La théorie des graphes, et plus généralement les mathématiques discrètes, constituent un domaine des mathématiques qui, historiquement, s'est aussi développé au sein de disciplines diverses telles que la chimie (modélisation de structures), la biologie (génome), les sciences sociales (modélisation des relations) ou en vue d'applications industrielles (problème du voyageur de commerce). Ces dernières années, la théorie des graphes a connu un regain d'intérêt pour ses capacités à modéliser de nombreux problèmes d'informatique. En effet, un graphe peut modéliser un programme, un algorithme, mais aussi l'architecture d'un ordinateur. Nous nous limiterons dans ce texte à l'exemple des réseaux d'interconnexion d'architecture parallèles.

Schématiquement, une machine parallèle est constituée d'un ensemble de processeurs élémentaires qui peuvent communiquer entre eux via un réseau. Dans les machines à mémoire distribuée, un processeur est composé d'une unité centrale, d'une mémoire locale et d'un routeur qui permet de transmettre les messages sur des canaux. Le temps pris par les communications entre processeurs, qui ont lieu à travers le réseau d'interconnexion, est un paramètre important pour les performances des machine parallèles.

Le réseau d'interconnexion est modélisé par un graphe dont les sommets représentent les processeurs et les arêtes les canaux. La figure 1 montre des exemples de réseaux classiques :
- le graphe complet, dans lequel chaque processeur peut communiquer directement avec tous les autres processeurs ;
- l'anneau (ou cycle), très utilisé pour relier par exemple des stations de travail dans un réseau local ;
- l'hypercube, qui est le réseau utilisé dans la *Connection Machine* de Thinking Machine Corporation (CM2 à 2^{16} processeurs), ou dans l'iPSC de chez Intel ;
- la grille 2 ou 3-dimensionnelle (éventuellement torique), à la base des nouvelles machines d'Intel (iWarp, Paragon), de Fujitsu (AP1000)...

Les briques de base actuellement utilisées pour construire ces réseaux sont des Transputers (Inmos, T800 ou futurs T9000), des processeurs Sparcs, des processeurs Motorola ou des i860 (Intel).

Faciliter la communication entre processeurs

Le réseau d'interconnexion idéal doit permettre à de nombreux processeurs de communiquer de manière facile, efficace et sûre, tout en satisfaisant diverses contraintes :
- Un processeur ne peut être connecté qu'à un nombre limité d'autres processeurs, ce qui se traduit dans le graphe par un degré maximum borné. Le modèle actuel du transputer a un degré de 4.
- Deux nœuds quelconques doivent pouvoir communiquer rapidement. Si les routeurs utilisent la technique dite *Store-and-Forward*, où un message est stocké avant d'être transmis au nœud suivant, le temps de communication entre deux nœuds est proportionnel à la distance entre les sommets correspondants du graphe (longueur d'un plus court chemin). On s'intéressera donc au diamètre du graphe, qui est la distance maximum entre toute paire de sommets du graphe.
- Les chemins sur lesquels les messages sont acheminés doivent pouvoir être déterminés de manière simple.
- Les liens doivent être courts, de manière à couvrir une grande bande passante (grand débit de communication).
- Le réseau doit être résistant aux pannes, c'est-à-dire que si certains processeurs ou liens sont défectueux, les messages doivent toujours être acheminés entre les processeurs restants. Cela se traduit sur le graphe par une grande connexité. On dit qu'un graphe est k-connexe si le graphe obtenu en supprimant au plus k-1 sommets reste connexe (c'est-à-dire qu'il existe encore un chemin entre toute paire de sommets). D'après le théorème de Menger, classique en théorie des graphes, "un graphe est k-connexe si et seulement s'il existe k chemins sommets-disjoints entre toute paire de sommets du graphe". Dans la pratique, on veut que ces chemins ne soient pas trop longs, ce qui a donné naissance à l'étude d'un paramètre nouveau qui mesure le diamètre maximum du graphe restant en cas de panne de sommets ou d'arêtes. L'existence de plusieurs chemins est utile, même en l'absence de panne, pour transmettre un long message en le découpant en paquets et en envoyant chaque paquet séparément sur chacun des chemins.

Il est impossible de satisfaire toutes ces contraintes en même temps, ce qui rend la conception du réseau difficile et nécessite de disposer d'une étude théorique poussée pour comparer les divers paramètres, et permettre de faire un choix en toute connaissance de cause.

Une des familles ayant un nombre maximum de sommets pour un diamètre et un degré fixé est celle des réseaux dits de de Bruijn (Fig. 2). A notre connaissance, il n'existe pas de machines parallèles commerciales les utilisant, mais certaines machines à base de Transputers sont interconnectées suivant ce modèle, et un décodeur employant ce type de réseau est utilisé dans le projet Galileo à Caltech (USA). De nombreuses études ont été faites pour déterminer leurs propriétés encore loin d'être complètement élucidées.

Ces recherches, et plus généralement, l'étude des problèmes de communication dans les réseaux, sont menées en France dans le cadre du groupe RUMEUR du PRC GDR C3 qui rassemble des équipes d'Orsay (LRI), de Nice-Sophia Antipolis (I3S), de Lyon (LIP), de Grenoble (IMAG), de Bordeaux (LABRI), et de Rennes (IRISA). D'autres études sont en cours pour trouver comment plonger au mieux dans un graphe G de processus (ou un réseau de processeurs) dans un réseau H donné, de manière à ce que H simule efficacement toutes les communications spécifiées par G. Toutes font l'objet de programmes internationaux.

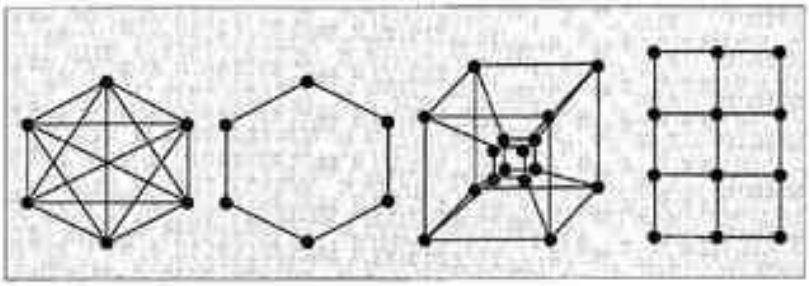


Fig. 1 - Un graphe complet, un anneau, un hypercube et une grille.

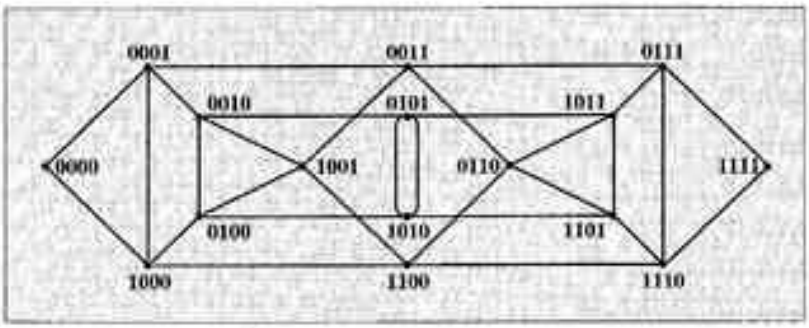


Fig. 2 - Un réseau de de Bruijn à 16 sommets.

Jean-Claude Bermond, directeur de recherche au CNRS, directeur du Laboratoire d'Informatique, signaux et systèmes I3S (URA 1376 CNRS).
Emmanuel Lazard, allocataire normalien BFR, Laboratoire de recherches en informatique LRI (URA 410 CNRS).
Dominique Sotteau, chargée de recherche au CNRS, LRI (URA 410 CNRS), Bât. 490, Université Paris-Sud, 91405 Orsay Cedex.
Michel Syska, allocataire MRT, Laboratoire I3S (URA 1376 CNRS), 250, avenue Albert Einstein, Sophia-Antipolis, 06560 Valbonne.

LA RECHERCHE OPÉRATIONNELLE

La recherche opérationnelle a pour but d'aider à trouver des solutions concrètes aux problèmes complexes que posent les grands systèmes d'hommes ou de machines.

Catherine Roucairol

La recherche opérationnelle utilise des méthodes scientifiques pour guider la prise de décisions face à des problèmes complexes que l'on rencontre dans la gestion et la direction de grands systèmes d'hommes, de machines, de matériaux et d'argent dans l'industrie, le commerce, l'administration ou la défense. Plutôt qu'une approche quantitative de ces problèmes, elle propose une véritable démarche scientifique pour les aborder : chercher à comprendre et aider à structurer une situation complexe, utiliser cette compréhension pour prédire son évolution et améliorer ses performances, en tirant le meilleur parti des moyens dont on dispose dans des circonstances données.

Discipline carrefour, elle utilise pour arriver à ses fins des techniques analytiques et numériques propres à d'autres disciplines scientifiques : statistique (recueil des données), économie, systémique (compréhension, analyse de l'environnement en vue de modèles), mathématiques continues ou discrètes, et, bien sûr, informatique (élaboration d'une méthode et traitement).

Organiser la tournée d'un livreur

Pour illustrer un de ses aspects, examinons un problème concret. Un livreur à bord d'un camion se rend chaque matin dans les magasins de clients pour décharger ses marchandises, avant de retourner à son point de départ, le dépôt central. L'entreprise cherche à organiser sa tournée. Plus elle est courte (en temps ou kilomètres parcourus), plus elle sera économique pour l'entreprise. Cette analyse du problème, bien qu'ici un peu caricaturale par sa rapidité, définit un modèle du problème. Le nombre de tournées possibles correspond au nombre de choix possibles pour le premier client, multiplié par le nombre de ceux pour le second client, puis par celui de ceux du troisième, et ainsi de suite jusqu'au seul choix du dernier, soit $25\times24\times23\times\ldots\times2\times1$ possibilités pour 25 clients. Mais la tournée cherchée ne pourra être trouvée par énumération. Même un ordinateur calculant assez rapidement (disons additionnant 25 nombres en 1 microseconde) utiliserait un temps équivalent à environ 2,5 milliards de siècles pour énumérer tous les cas donnés par ce nombre à 24 chiffres.

Pour dépasser cette "explosion combinatoire", la recherche opérationnelle (RO), a développé un certain nombre d'outils. Ainsi, s'agissant de tournée, quoi de plus naturel que de s'aider du tracé des routes reliant les clients ? Dans ce dessin ou graphe, la tournée est représentée par un circuit, partant du dépôt et y revenant, passant une fois et une seule par chaque client (point ou sommet du graphe) : un circuit dit "hamiltonien". La théorie des graphes, inventée par Claude Berge en 1958, caractérise plusieurs structures reconnaissables, et offre de nombreuses méthodes ou algorithmes de résolution. Plus que toute autre discipline, la RO résulte d'une féconde confrontation entre la théorie et la pratique : devant l'importance de certains problèmes traités, de nouvelles techniques mathématiques (programmation mathématique, processus stochastiques, optimisation combinatoire) ont été inventées ou developpées (théorie des graphes, théorie des jeux). La généralisation des micro-ordinateurs et des stations de travail en a facilité l'emploi et banalisé les méthodes. La plupart de ces techniques mathématiques sont disponibles sous forme de logiciels commercialisés.

Vers des solutions approchées satisfaisantes

Mais la RO ne se limite pas à l'utilisation de ce noyau dur d'outils dans une séquence modélisation-résolution-mise en œuvre, elle se veut de plus en plus une aide à la décision. Lorsque les critères d'optimisation sont multiples et non hiérarchisés, des méthodes dites interactives (système interactif d'aide à la décision ou SIAD), voire des systèmes experts lui permettent de contrôler la valeur des décisions et le degré de perfection des solutions obtenues. Elle peut aussi abandonner la recherche d'une solution optimale au profit plus réaliste de solutions approchées satisfaisantes (dites "heuristiques"), en s'appuyant sur des méthodes empruntées à d'autres disciplines (recuit simulé emprunté à la thermodynamique, méthode tabou à l'intelligence artificielle, algorithmes génétiques ou neuronaux à la bioinformatique).

OPERATIONAL RESEARCH - Operational researchers aim to provide rational bases for decision making: they seek to understand and structure complex situations and to use this understanding to predict system behaviour and improve system performance.

On l'aura compris, la RO ne peut résoudre tous les maux de l'humanité, mais elle joue un rôle important dans les secteurs public et privé en aidant à comprendre les objectifs et valeurs mises en jeu dans des problèmes de plus en plus complexes. En France, ses méthodes sont couramment et depuis longtemps employées par les pétroliers (programme de production), les entrepreneurs de construction (planification des chantiers, célèbre méthode PERT), les fabricants alimentaires (mélange au moindre coût, gestion prévisionnelle des stocks). Des transporteurs comme Air-France, la RATP, la SNCF établissent la rotation des équipages sur les avions, l'habillage horaire des lignes de métro, la marche des trains grâce à ses modèles.

Des sociétés de conseil aident les entreprises grandes ou petites à élaborer des programmes de production, gérer des achats, concevoir des tournées (distribution, ramassage), localiser des entrepôts, mieux dessiner des fuselages d'avion, établir des plans de découpe avec des chutes minimales dans les aciéries, verreries ou usines de confection, gérer les stocks de boissons suivant les prévisions météorologiques…

Même l'informatique pour être efficace a besoin de la RO. Pour preuve, l'ordonnancement ou ordre de passage de travaux sur un ordinateur ou un réseau, l'évaluation des performances (des résultats puissants ont été obtenus grâce à la théorie des files d'attente), la gestion des bases de données (temps d'accès, taille d'une requête), le placement et le routage des circuits intégrés des VLSI en CAO, la conception et la construction de nouvelles machines parallèles (accélération des compilateurs détectant le parallélisme d'un programme), le placement de processus, c'est-à-dire d'instructions d'un programme sur des processeurs parallèles pour des communications minimales, le routage rapide des messages dans un réseau distribué.

Catherine Roucairol, professeur à l'Université de Versailles-Saint Quentin et à l'INRIA, Unité Méthodologie et architecture des systèmes informatiques (URA 818 CNRS), Université de Versailles-Saint Quentin, 45, avenue des Etats-Unis, 78000 Versailles.

LE CALCUL FORMEL

L'ordinateur n'est pas limité à la manipulation des nombres, il est aussi capable de calculs formels en travaillant sur des objets mathématiques comme les polynômes, les fonctions trigonométriques, les groupes, les idéaux, les tenseurs et les systèmes différentiels.

FORMAL CALCULUS - Computers are also able to handle symbols and therefore solve mathematical problems involving elementary functions. The obstacle of prohibitive cost has disappeared as prices of components fall. Formal methods are therefore likely to gain ground in many areas of application for robotics, chemistry, signal processing and even biology.

Jean Della Dora

Les systèmes de calcul formel, capables de manipuler efficacement des symboles mathématiques, ont été créés dès 1956. Depuis cette date de très nombreux systèmes ont été développés. Un système universitaire comme REDUCE a été implanté sur 2000 sites (1/3 au Japon, 1/3 en Europe, 1/3 aux USA) et le nouveau système AXIOM, produit initialement par la société IBM, va accéder au niveau de système "industriel" en étant distribué par SAG dès 1991. Un premier frein au développement du calcul formel, grand consommateur de mémoire et de capacité de calcul, a été levé à la fin des années 1980 par l'explosion de la technologie (décroissance du prix des mémoires). Mais il en reste un, de nature intellectuelle. Pour aborder de nouveaux champs prometteurs d'applications, il faut développer une collaboration étroite entre mathématiques et informatique. Or il est difficile de trouver des chercheurs qui soient simultanément en pointe dans les deux disciplines.

Des polynômes...

La complexité de certains algorithmes de manipulation polynomiale (calcul du PGCD de deux polynômes) a été clairement mise en évidence avant 1980. Depuis, de nombreux progrès ont été réalisés, par exemple sur les systèmes de valeurs de variables qui annulent simultanément une famille de polynômes. De même, les travaux des vingt dernières années ont résolu en grande partie le problème complexe de la factorisation des polynômes.

Domaine privilégié de nombreux mathématiciens, la géométrie algébrique constructive a connu un immense développement depuis 1965, date de la découverte par B. Buchberger d'un algorithmes simple et bien adapté au problème. Le calcul symbolique a beaucoup étudié cet algorithme (surtout depuis 1975), et les travaux actuels de l'école française (D. Lazard, A. Galigo, M. Giusti) sont des références internationales.

...aux équations différentielles

Le calcul des solutions d'une équation différentielle sous la forme de fonctions élémentaires, considérées comme des éléments d'un corps obtenu en itérant un processus de construction symbolique – prise d'exponentielle ou de logarithme de fonctions rationnelles ou algébriques – est un autre domaine important. Le calcul des primitives a connu des progrès considérables tant mathématiques que symboliques ces dernières années. Depuis les travaux de Liouville (1833) jusqu'à la thèse de M. Bronstein (1987), en passant par les contributions décisives de l'école anglaise (J. Davenport, A. Norman) et de l'école américaine (B. Trager), on est passé du calcul de l'intégrale de fraction rationnelle jusqu'à un algorithme (Bronstein) montrant que l'on peut décider si la primitive d'une fonction élémentaire est élémentaire, et dans l'affirmative la calculer. Précisons un point essentiel : il s'agit bien d'algorithme et non d'heuristique. Le cas le plus simple de solution d'équations différentielles concerne les systèmes de la forme :

$x^q y' = A(x)\, y$ où $A(x)$ est une matrice d'ordre n à coefficients polynomiaux et q un nombre entier.

Les années 80-90 ont permis de créer des algorithmes efficaces pour résoudre formellement de tels systèmes, c'est-à-dire former autour de la singularité zéro, un système fondamental de solutions (ces solutions sont exprimées par des séries dont les coefficients sont exacts). Malheureusement, ces séries sont en général divergentes. La sommation de ces séries est un très difficile problème mathématique qui commence à être résolu algorithmiquement et mathématiquement grâce aux efforts d'un groupe de mathématiciens (B. Malgrange, J. Martinet, J.P. Ramis, J. Ecalle).

Signalons enfin que les travaux sur les systèmes non linéaires avancent rapidement, mais que de très grandes difficultés mathématiques obstruent pour le moment une solution algorithmique.

Les applications se multiplient

Ces études n'ont pas que des objectifs fondamentaux et désintéressés. Par exemple, Delaunay a mis dix ans pour calculer (à la main...) l'orbite de la lune, et dix autres années pour la vérifier. Le calcul n'est pas numérique (il demande le traitement de formules qui occupent 128 pages). Ces calculs sont faits actuellement en quelques heures.

Le calcul symbolique trouve actuellement dans le domaine de la chimie un très grand champ d'application. Un problème typique est celui de la prédiction du comportement d'une molécule (réactions chimiques). Les modèles de comportements sont très complexes, les plus récents ont conduit à des calculs symboliques, puis à la production automatique des codes numériques. Le problème de la détermination pour un robot des angles de rotation et des translations des joints permettant au bras de venir dans une position donnée conduit à des systèmes d'équations algébriques qui sont solubles par les méthodes précédentes.

Le traitement du signal et le codage, la théorie du contrôle, la biologie mathématique, la théorie des groupes sont autant de domaines où le calcul formel a déjà donné d'excellents résultats.

On ne peut donc plus faire au calcul formel le procès de son applicabilité. Nous pensons même que le calcul formel montre la voie aux futurs systèmes intégrés scientifiques de demain et qu'il sera probablement incontournable.

Jean Della Dora, professeur à l'Institut national polytechnique de Grenoble, directeur du Centre de calcul de Grenoble, Laboratoire de modélisation et calcul (URA 397 CNRS), 46, avenue Félix Viallet, 38031 Grenoble.

GÉOMÉTRIE ALGORITHMIQUE

Les ordinateurs ont maintenant de plus en plus souvent à résoudre des problèmes de nature géométrique. La réduction des temps de traitement est une priorité absolue dans l'écriture des algorithmes.

COMPUTATIONAL GEOMETRY - Data processing increasingly involves geometric problems for which efficient algorithms have to be found. Voronoi diagrams and Delaunay triangulation are typical tools. Motion planning for robots may be reduced to the description of the collision free space, i.e. all positions of the robot that are collision-free.

Jean Berstel
Michel Pocchiola

De très nombreuses applications font intervenir des problèmes de nature géométrique : la conception de circuit VLSI demande des algorithmes sur les rectangles ; en infographie, on manipule constamment des objets géométriques ; la planification des trajectoires en robotique fait également appel à la géométrie.

Dans la pratique, le nombre d'objets à traiter est en général important (des dizaines de milliers de facettes dans une image réaliste par exemple); le développement d'algorithmes et de structures de données sophistiqués est donc rentable s'il permet de réduire le temps de traitement et les capacités de mémoire nécessaires. Remplacer un algorithme qui traite le problème en un temps proportionnel à n^2, par un algorithme en $n \log_2 n$ divise le temps de calcul par 100 si $n = 1000$. Passer à un algorithme en n permet de gagner encore un facteur 10.

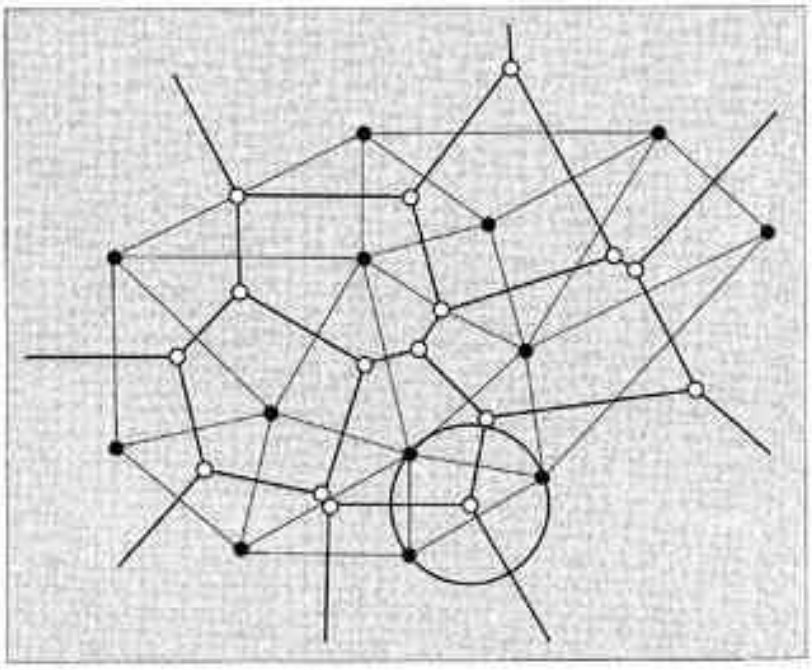
Les solutions intègrent des concepts et des méthodes d'horizons divers : théorie des graphes, géométrie combinatoire, calcul des probabilités, théorie des nombres... Cette diversité conduit à un ensemble d'algorithmes efficaces et sophistiqués, dont la preuve et l'analyse sont toutefois souvent difficiles.

Le bureau de poste le plus proche

Considérons par exemple le problème suivant, connu sous le nom du *problème des bureaux de poste* : on se donne un ensemble de n points (les bureaux de poste ou les sites figurés par des points noirs dans la figure) ; étant donné un nouveau point p, le problème est de déterminer le bureau de poste le plus proche. On appelle requête une telle demande. Une solution simple à ce problème est de calculer la distance de p à chacun des sites n et de retenir celui qui réalise le minimum. Ceci requiert un temps linéaire en n. Considérons maintenant la situation où le bureau de poste le plus proche doit être calculé pour beaucoup d'expéditeurs, en d'autres termes, où le nombre de requêtes est important. Dans ce cas, un prétraitement des sites peut diminuer le temps de réponse par requête. Dans notre exemple, on connaît des prétraitements en temps proportionnel à $n \log n$ qui permettent de répondre à chaque requête en temps logarithmique.

Que se passe-t-il si l'administration des postes décide de supprimer ou déplacer un bureau de poste, voire d'en créer ? La gestion dynamique efficace des structures de données représentant les objets géométriques constitue un problème fondamental. De nombreux problèmes géométriques se présentent ainsi : recherche d'une structure de données adaptée à l'exécution efficace de requêtes et à la gestion dynamique des objets eux-mêmes.

Revenons au problèmes des bureaux de poste. Pour tout site s, l'ensemble des points du plan qui sont plus proches de s que de tout autre site est un polygone convexe $V(s)$ appelé la *région de Voronoï* (sur la figure, les contours des polygones sont en traits épais et leurs sommets, des cercles blancs). La partition du plan induite par ces polygones est appelée le *diagramme de Voronoï* en l'honneur du mathématicien polonais G. Voronoï. Les applications fondées sur l'usage des diagrammes de Voronoï sont nombreuses, en géométrie algorithmique, en statistique, en géographie, en chimie, et en bien d'autres domaines. Le diagramme de Voronoï a de nombreuses propriétés, notamment son dual géométrique, où les sommets des polygones de Voronoï deviennent des sites, est une triangulation des sites appelée la *triangulation de Delaunay* (voir sur la figure, où les triangles sont en traits fins). Elle est caractérisée par la propriété suivante : les cercles



Un diagramme de Voronoï

circonscrits aux triangles sont exactement les cercles passant par trois sites et ne contenant aucun autre site.

Un exemple d'application est le calcul d'un arbre connectant les sites et dont la somme des longueurs des arêtes est minimale. On appelle un tel arbre un *arbre recouvrant euclidien minimal* et on peut montrer qu'il est inclus dans la triangulation de Delaunay.

Il existe plusieurs algorithmes optimaux pour calculer le diagramme de Voronoï ; certains utilisent une méthode dichotomique qui procède récursivement : on partage l'ensemble des sites en deux parts égales, on calcule séparément les diagrammes des deux parties, puis on "recolle" les diagrammes. Un autre algorithme utilise le paradigme du balayage : une ligne (fictive) balaye le plan à la rencontre des sites. Chaque rencontre crée un ou plusieurs "événements" qui permettent de construire au fur et à mesure les sommets et arêtes du diagramme.

La structure obtenue permet de déterminer la région de Voronoï à laquelle appartient un point donné en temps logarithmique. Il existe des algorithmes de gestion dynamique d'un diagramme de Voronoï qui sont logarithmiques en moyenne.

Les gardiens de musée

Les problèmes de visibilité, comme l'élimination des lignes et des faces cachées dans la fabrication des images de synthèse sont à la source d'un ensemble important de travaux en géométrie algorithmique.

Un des plus anciens exemples est le problème dit de la *galerie d'art*, posé la première fois par V. Klee : "Quel est le nombre minimal de gardiens qui suffisent pour surveiller une galerie d'art à n côtés ?". La réponse est étonnamment simple : $\lfloor n/3 \rfloor$ gardiens suffisent, et sont parfois nécessaires. Trouver le bon emplacement des $\lfloor n/3 \rfloor$ gardiens est plus difficile ; une méthode commence par construire une triangulation du polygone, et ce n'est que tout récemment que l'on a découvert un algorithme linéaire de triangulation (voir figure).

Le déplacement d'un robot

On appelle planification de trajectoire l'étude des déplacements possibles d'un objet géométrique (le robot) à l'intérieur d'une scène contenant d'autres objets (les obstacles). Parmi les questions posées figurent l'existence d'une trajectoire (sans collision), l'absence de collisions pour une trajectoire donnée, l'influence de la nature des mouvements autorisés (translations, rotations), la connaissance partielle ou globale que peut avoir le robot de son environnement, et bien entendu la sensibilité des trajectoires à une modification de l'environnement.

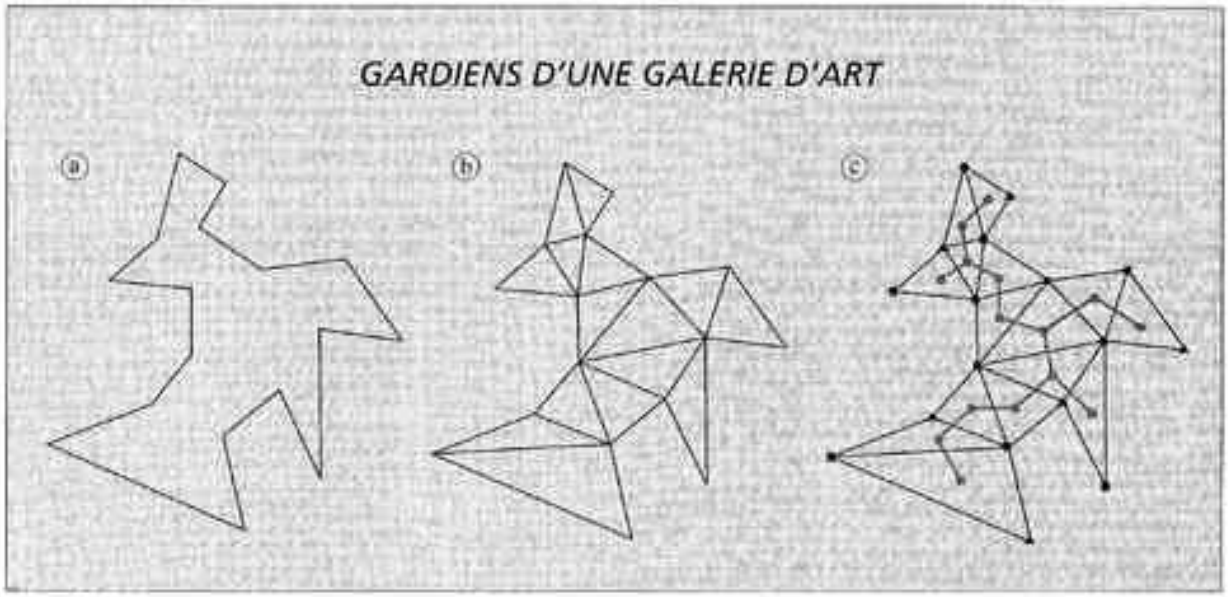
Il existe des méthodes très générales pour résoudre le problème de la planification de trajectoires, avec des contraintes très faibles sur la nature des objets en présence, dans le cas d'un environnement connu. Elles consistent à décrire l'espace libre, c'est-à-dire l'ensemble des positions du robot sans collision avec les obstacles.

Dans des cas particuliers, des méthodes simples et efficaces ont été développées. L'une d'entre elles est la méthode de *rétraction*. Ainsi, les trajectoires d'un disque dans un environnement polygonal peuvent être calculées à partir du diagramme de Voronoï des polygones. Pour déterminer une trajectoire, on rétracte le centre du disque sur le diagramme de Voronoï, puis on suit les arêtes du diagramme.

Ces quelques exemples mettent en évidence la diversité et souvent la difficulté des sujets traités. Les recherches actuelles portent principalement sur la précision des calculs, la gestion efficace des cas dégénérés, la prise en compte du mouvement des objets. L'échantillonnage aléatoire est une technique aux nombreuses applications qui fournit généralement des algorithmes plus simples au prix d'une analyse plus complexe.

Jean Berstel, professeur des universités, Laboratoire d'informatique théorique et programmation (URA 248 CNRS), Institut Blaise Pascal, 4, place Jussieu, 75252 Paris Cedex 05.

Michel Pocchiola, maître de conférences à l'École normale supérieure, Laboratoire d'informatique de l'ENS (URA 1327 CNRS), 45, rue d'Ulm, 75230 Paris Cedex 05.



La galerie d'art de la figure a est triangulée comme indiqué dans la figure b. La relation d'adjacence entre triangles, en grisé sur la figure c, est connexe et sans cycle. C'est cette propriété qui permet de colorer en trois couleurs les sommets de la triangulation. La couleur la moins fréquente (rouge) détermine un emplacement d'au plus $\lfloor n/3 \rfloor$ gardiens couvrant la galerie.

PROBABILITÉS ET INFORMATIQUE

Les probabilités interviennent selon deux modalités en informatique : pour modéliser une réalité fluctuante, ou en algorithmique quand une solution est soit impossible, soit trop difficile à atteindre.

PROBABILITIES AND INFORMATION TECHNOLOGY - Probabilities are used in data processing to compute the speed of algorithms and to evaluate information systems. But they are also used for applications algorithms when deterministic algorithms are too complex or too lengthy or else when the problem is fundamentally non-deterministic. These algorithms often use series of pseudo-random numbers the quality of which needs to be checked.

Brigitte Plateau

Les probabilités sont utilisées en informatique pour modéliser et évaluer le comportement d'un algorithme, mais aussi d'un système complet, architecture matérielle comprise, ou sont à la base d'algorithmes dits probabilistes.

L'art de calculer la rapidité d'un algorithme et la taille mémoire qu'il occupe s'appelle l'analyse d'algorithme. Un paramètre important de cette analyse est la taille des données et la configuration de ces données. Si toutes les configurations ont la même chance d'intervenir, on dira qu'elles sont équiprobables. Si certaines ont plus de chance d'intervenir que d'autres, la loi de probabilité devra le prendre en compte. Par exemple, lorsque l'on trie de nouveau une liste déjà triée mais qui a été perturbée, deux éléments consécutifs de la liste ont une probabilité supérieure à 0,5 d'être classés dans le bon ordre de départ. L'analyse d'algorithme consiste à faire l'hypothèse d'une loi de probabilité sur les données, et de calculer les moyennes, variances et autres caractéristiques des performances de l'algorithme.

Les systèmes informatiques utilisent des algorithmes de contrôle. Il s'agit de surveiller l'accès des ressources que sont les mémoires, les processeurs, les disques, les liens de communication. Les clients de ces ressources sont les programmes, les messages, etc. Le concepteur du système veut sélectionner, dans un contexte donné, le meilleur algorithme de contrôle. Certaines quantités sont déterministes, comme la vitesse du processeur; d'autres non, c'est le cas des demandes de mémoire du programme, ou de la durée de ce même programme. Des mesures sur le système en fonctionnement permettent d'identifier des lois de probabilités, permettant de calculer les temps de réponse moyens pour les clients, les taux moyens d'utilisation des ressources etc. Dans le même ordre d'idée, les études sur la fiabilité des logiciels offrent des méthodes pour prédire l'occurrence des pannes.

Les algorithmes probabilistes

Il existe également des cas où les probabilités interviennent dans les algorithmes d'application. Les algorithmes d'optimisation – par exemple le recuit simulé – servent à calculer l'extremum d'une fonction de plusieurs variables. Une méthode de gradient déterministe mènerait à l'extremum local le plus proche. Pour tenter d'atteindre l'extremum absolu, une méthode consiste à perturber la méthode du gradient par des mouvements aléatoires.

Dans cette catégorie se trouvent aussi certains algorithmes de la théorie des nombres. Il existe par exemple des variantes du test de primalité de Fermat, qui assurent le caractère premier d'un entier avec une probabilité d'erreur inférieure à 2^{-N} pour N itérations de l'algorithme. Cette probabilité peut donc devenir aussi petite que l'on veut.

De manière générale, les algorithmes probabilistes reposent sur l'une des idées suivantes : soit les algorithmes déterministes connus pour le problème sont trop complexes et une réponse tolérant une probabilité d'erreur est acceptée, soit il n'existe pas d'algorithme déterministe résolvant ce problème mais seulement des solutions probabilistes.

De nombreux algorithmes probabilistes utilisent des séries de nombres aléatoires. Diverses méthodes existent pour les engendrer. En fait, les séries ne sont jamais parfaitement aléatoires. Des tests statistiques sont utilisés pour évaluer leur qualité.

Brigitte Plateau, professeur à l'Institut national polytechnique de Grenoble, Laboratoire de génie informatique de Grenoble (URA 398 CNRS), IMAG, 46, avenue Félix Viallet, 38031 Grenoble Cedex.

Détection de contours par recuit simulé (approximation par champs moyens) appliquée à une image IRM (coupe de rein), d'après J. Zerubia, INRIA et R. Chellappa, University of Maryland. ©Cliché A. Edelman - INRIA.

L'intelligence artificielle

Les opérations que peut effectuer un ordinateur ne se limitent pas à des calculs numériques. Il peut aussi manipuler des symboles abstraits : les mémoriser, les structurer, les comparer, les associer en motifs. Il peut même les acquérir en interagissant avec son environnement, via divers appendices physiques de perception et de communication. Ces opérations sur des symboles ne sont-elles pas en définitive les briques de base suffisantes pour appréhender l'intelligence, tout au moins en partie ? Telle est la question qui motive les recherches en intelligence artificielle (IA).

La thèse sous-jacente est fructueuse du fait des débats et des développements qu'elle soulève dans de nombreux domaines. Elle peut être correctement défendue, du point de vue de l'informatique, en concevant des machines performantes pour des tâches qui requièrent clairement de l'intelligence. Ceci conduit à une double démarche : faire des machines utiles pour un spectre large de tâches, et éclairer les mécanismes cognitifs en développant des modèles de raisonnement et des programmes qui permettent l'expérimentation. L'IA n'est certes pas la réalisation en machine de l'intelligence humaine; néanmoins, elle contribue par sa démarche à l'étude de cette faculté de l'homme.

L'objet central de l'IA concerne donc la manipulation de symboles, que nous appellerons des "connaissances". Il s'agit de les représenter sur ordinateur, de mécaniser des raisonnements les manipulant et de les acquérir, éventuellement, par apprentissage. Il existe de nombreuses représentations des connaissances qui ont des pouvoirs d'abstraction variés et sont associées à diverses formes de raisonnement. L'IA étudie notamment :

- la déduction, qui explicite des conséquences logiques à partir de schémas d'inférence du type : *de A, et de A implique B, on déduit B* ;
- l'abduction, qui conjecture les causes plausibles d'une observation *B*, sachant que *A cause B*, ces causes ne sont pas logiquement déductibles des connaissances disponibles ;
- l'induction, à la base de l'apprentissage, qui recherche des lois généralisant un ensemble de faits observés ou des concepts décrivant des objets regroupés comme "proches" ;
- l'analogie, qui rapproche deux situations reconnues comme similaires, pour envisager l'application à l'une d'elles de l'homologue d'une conclusion qu'on sait être vraie pour l'autre ;
- le raisonnement par défaut qui, à partir d'informations incomplètes, infère des conclusions plausibles ;
- la révision et le maintien de la cohérence des connaissances, suite à une évolution du monde ou à l'arrivée de nouvelles informations.

Par ailleurs, certains types de raisonnements sont spécifiques aux concepts particuliers qu'ils manipulent. Ce sont par exemple :
- le raisonnement sur le temps, ou sur les relations spatiales et géométriques ;

- le raisonnement sur les actions et leurs effets, et plus généralement sur le changement ;
- le raisonnement sur les connaissances et les croyances, prenant en compte ce qu'un agent sait ou croit savoir des connaissances d'un autre agent ;
- le raisonnement décisionnel : utilité et pondération des différents éléments d'un choix, formation de consensus, etc.

Un raisonnement doit pouvoir être explicité. C'est le problème de l'explication des conclusions obtenues par un système. Le problème du raisonnement sur le raisonnement, outre son intérêt pour l'explication, est également lié à la capacité de reformuler ses propres connaissances (voire ses méta-connaissances).

L'IA s'intéresse également à des approches numériques qui n'ont pas besoin d'une représentation explicite des connaissances, accessible à l'utilisateur, ni d'une articulation détaillée du raisonnement. C'est le cas des approches connexionnistes qui sont à considérer, à côté des approches symboliques et statistiques, pour des problèmes tels que la classification ou l'apprentissage.

La mécanisation de ces différents types de raisonnements conduit à des représentations et à des modes d'inférence dépassant le cadre de la logique classique : logiques dites non-monotones, logiques de l'incertain, logiques de la révision, calcul qualitatif, réseaux connexionnistes, logique floue de la gradualité, etc.

En amont du processus d'inférence proprement dit, existent des problèmes d'acquisition des connaissances, de vérification de leur cohérence (voire de leur complétude), et de leur compilation en procédures efficaces. Ces problèmes conduisent à de nouvelles préoccupations algorithmiques et à la conception de nouveaux logiciels spécialisés.

Plus générale que la problématique de l'acquisition des connaissances, celle de l'apprentissage automatique cherche à améliorer les performances de la machine et à étendre ses connaissances par l'étude d'exemples ou d'interactions avec l'environnement.

En dehors des enjeux purement scientifiques et intellectuels liés aux nouveaux problèmes soulevés, les recherches en IA correspondent à des applications existantes ou potentielles importantes. Au niveau des sytèmes d'inférence, on peut citer comme grands champs d'applications :
- la surveillance, le diagnostic ou l'aide à la maintenance. Ainsi dans le domaine médical, parmi de très nombreux systèmes, on peut citer par exemple ONCOCIN (suivi et traitement du cancer) ou PUFF (maladies pulmonaires) qui supportent une utilisation clinique quotidienne ;
- la gestion de production, la planification et le contrôle d'exécution de tâches : gestion d'un atelier ou d'un réseau électrique, gestion des actions d'un robot mobile ;
- le contrôle de processus complexes : circulation aérienne ou autoroutière, industrie chimique ou sidérurgique. Un bon exemple ici est l'ambitieux projet SACHEM d'aide au contrôle des hauts fourneaux de SOLLAC ;
- l'aide à la conception ou à la configuration de machines, de réseaux, de systèmes de communications, mais aussi d'ateliers : par exemple le système XCON de configuration d'ordinateurs apporte à l'industriel DEC une économie très substantielle ;
- l'aide à la traduction, à la génération et au traitement de textes en langue naturelle ;
- l'enseignement assisté, adapté au profil de l'utilisateur et à ses progrès ;
- le contrôle d'appareils divers, avec une flexibilité et une ergonomie très poussées, par exemple dans tous les domaines grand public.

Les systèmes experts, qui ne recouvrent qu'une partie de ces applications, correspondent d'ores et déjà à un marché important : 500 millions de dollars pour les USA et l'Europe en 1990.

Les systèmes d'inférence que propose l'IA seront à plus long terme au cœur de machines intelligentes, intégrant des moyens de perception, d'action et de communication avec leur environnement. De telles machines recouvrent d'autres enjeux sociaux-économiques importants : exploration spatiale, exploration sous-marine, et plus généralement travaux pénibles, dangereux ou dans des environnements hostiles à l'homme.

Notre question initiale, sur les manipulations symboliques comme briques de base de l'intelligence, suscite moult réponses sceptiques : les processus mentaux ne sauraient être appréhendés par une décomposition réductionniste, fut-elle à base de réseaux neuronaux artificiels. Etant non décomposable, l'intelligence ne pourrait être dupliquée par une machine.

Comme la preuve de la marche en marchant, la réponse ne peut être ici qu'empirique, par la construction éventuelle de telles machines. Mais ce débat, aujourd'hui de nature philosophique, restera ouvert longtemps encore.

En attendant, l'IA contribue à perfectionner et à rendre plus puissants nos artefacts, lesquels comportent d'ores et déjà, et de plus en plus, des processeurs informatiques.

Malik Ghallab, directeur de recherche au CNRS.
Henri Prade, directeur de recherche au CNRS.

LA REPRÉSENTATION DES CONNAISSANCES

Les connaissances humaines exprimées en langage naturel ne sont pas écrites pour être comprises par les ordinateurs. Il est préférable de les coder et de les traiter selon des modalités logiques particulières.

KNOWLEDGE REPRESENTATION – An "intelligent" program must be able to process a wide range of highly varied, often poorly or incompletely defined knowledge. New forms of logic capable of using this so-called common-sense knowledge have to be devised and new modes of representation need to be found.

Daniel Kayser

Pour le moment, l'utilité des programmes d'intelligence artificielle dépend principalement de la qualité des connaissances que les êtres humains leur fournissent. Les programmes d'apprentissage ne font pas exception, il faut leur apporter beaucoup de connaissances (humaines) pour qu'ils puissent acquérir quelques nouvelles connaissances utilisables.

Les connaissances humaines sont exprimées naturellement sous forme langagière, ou parfois graphique (figures, illustrations, cartes,...). Celles que peuvent exploiter les machines sont par contre des algorithmes ou des données. A moins de disposer d'algorithmes généraux de compréhension du langage ou des images, ce qui nous renvoie à un futur non spécifié, les informaticiens ne peuvent se contenter de "coder" les connaissances humaines : ils doivent représenter ces connaissances afin de les mettre à la disposition des machines.

Différentes formes de représentation sont envisagées ; comme en programmation, la tendance actuelle est de trouver des modalités proches des conceptions humaines, quitte à utiliser l'ordinateur pour transformer ces premières données en codes plus directement utilisables pour lui. C'est ainsi que se développent depuis une quinzaine d'années des langages spécialement consacrés à la représentation des connaissances.

L'adaptation au problème

On peut distinguer à l'heure actuelle plusieurs voies de recherche :

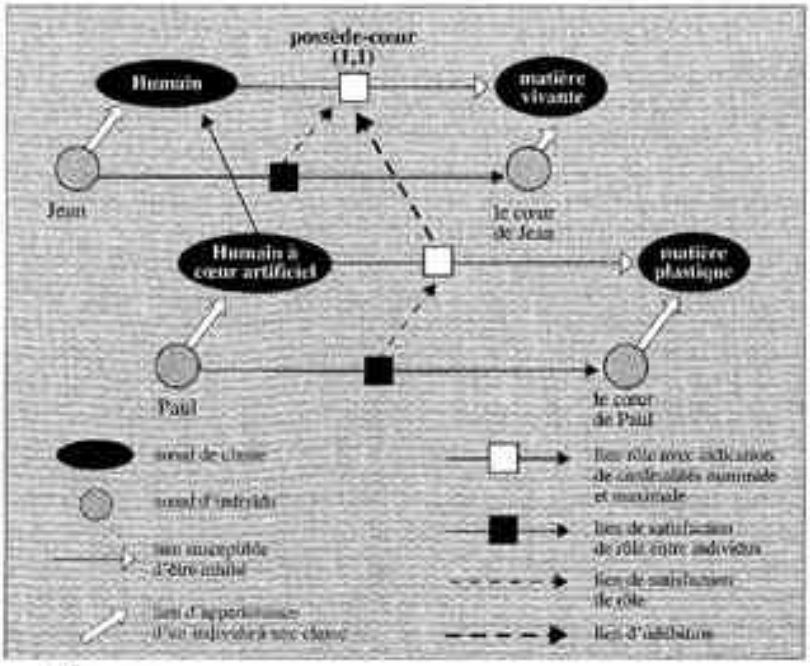
- lorsque les connaissances à représenter consistent en longues listes dont chaque membre a une structure comparable (par exemple : *Bogota est la capitale de la Colombie, Ottawa est la capitale du Canada,...*), les bases de données fournissent un cadre conceptuel et des techniques de saisie, de stockage et de traitement bien adaptées ;
- lorsque les connaissances à représenter correspondent à la mise en œuvre de principes déterminés afin d'aboutir, par exemple, à un diagnostic ou à une recommandation d'action, plusieurs cas peuvent se présenter.

Si les principes auxquels on se réfère sont spécifiques, si la causalité sous-jacente est mal connue ou si, parfaitement connue, elle implique des calculs trop complexes au regard du temps de réaction souhaité, on préfère substituer aux calculs des méthodes approximatives simplistes : la solution généralement adoptée est la représentation sous forme de règles de production. On utilisera alors un système expert.

Si les principes auxquels on se réfère sont généraux et s'il est possible, au moyen de calculs relativement simples, de résoudre exactement les problèmes que les utilisateurs sont susceptibles de se poser, les connaissances s'exprimeront sous forme d'algorithmes de simulation (par exemple, représentation de connaissances physiques) ou de formules logiques fournies à un démonstrateur de théorèmes (par exemple, représentation de connaissances mathématiques).

Un cas intermédiaire concerne les situations où les principes, bien que connus, mènent à des calculs trop compliqués, mais où ce qui intéresse l'utilisateur est de nature qualitative plutôt que quantitative. Il est alors possible de trouver des méthodes de résolution exactes qui fournissent en des temps acceptables les informations désirées. La représentation de l'évolution qualitative des systèmes a donné lieu au cours des dernières années à certains des travaux les plus originaux en intelligence artificielle.

Les cas les plus intéressants du point de vue informatique – et ces cas correspondent vraisemblablement aux formes les plus répandues des connaissances humaines – sont ceux où les connaissances à représenter sont de natures très hétérogènes, mettant en œuvre des concepts difficilement définissables : tout le monde sait que les jeunes enfants sont généralement turbulents mais peut-on définir rigoureusement la turbulence, l'âge auquel un enfant cesse d'être un ▶



Étude d'un réseau sémantique avec gestion des exceptions, interprétation logique et implantation informatique (d'après P. Coupey, thèse de l'Université Paris-Nord, janvier 1989).

jeune enfant, les circonstances où l'adverbe *généralement* est approprié ? On a souvent prétexté de ces difficultés pour décréter que ce type de connaissance n'était tout simplement pas représentable en machine, mais il est de fait que nous, humains, exploitons en toute occasion des connaissances de ce type, et la représentation des connaissances dites de *sens commun* est donc une voie de recherche qui n'a aucune raison a priori de conduire à une impasse.

Inventer de nouvelles logiques

Des résultats théoriques et pratiques importants ont d'ailleurs été obtenus dans cette voie au cours de la dernière décennie. Sur le plan théorique, les logiques non-monotones (c'est-à-dire celles dont l'ensemble des formules prouvables peut décroître lorsque l'ensemble des prémisses s'accroît) apportent des abstractions utiles pour gérer les situations où la connaissance se présente sous forme de normes plutôt que de règles infaillibles. Sur le plan pratique, les réseaux sémantiques et les outils logiciels qui permettent de les manipuler autorisent la représentation de connaissances hétérogènes sous des formes qui semblent relativement claires et utilisables pour les développeurs de systèmes informatiques "intelligents".

La problématique de la représentation des connaissances, qui a mis un certain temps à émerger des recherches ponctuelles menées dans différents domaines, a acquis depuis plus de dix ans une place centrale en intelligence artificielle ; cette place lui est depuis peu contestée et la possibilité d'une modélisation de l'intelligence sans représentation a été évoquée, notamment à l'occasion de réflexions sur l'auto-organisation et sur le connexionnisme. De telles positions sont cependant perçues actuellement comme "extrémistes" et l'avenir verra probablement une synthèse entre les approches à base de représentation et les techniques qui leur sont opposées.

Daniel Kayser, professeur à l'Université Paris-Nord, Laboratoire d'informatique de Paris-Nord (URA 1507 CNRS), Institut Galilée, Université Paris-Nord, avenue Jean-Baptiste Clément, 93430 Villetaneuse.

LE RAISONNEMENT DANS LES SYSTÈMES À BASES DE CONNAISSANCES

Les systèmes à bases de connaissances font appel à des types de raisonnement très variés qui sortent souvent de la logique classique. Cette diversité permet de mieux rendre compte de la grande richesse des processus cognitifs de l'être humain.

Jean-Paul Haton

REASONING IN KNOWLEDGE-BASED SYSTEMS – Artificial intelligence makes use of several forms of reasoning. The key issue is to conduct a sequence of operations on symbolic structures while ensuring that the conclusions are true and coherent. The tendency today is to combine several types of reasoning and knowledge.

Le raisonnement dans les systèmes à bases de connaissances est multiforme. Un dénominateur commun est le concept d'hypothèse, parcelle de connaissance nouvelle, créée par un mécanisme de raisonnement pour rendre compte d'un phénomène observé dans le cadre d'un univers d'application. La nature hypothético-déductive du raisonnement humain est une donnée fondamentale, bien connue des psychologues et des philosophes, et la manipulation d'hypothèses joue un rôle central en intelligence artificielle (IA).

Un raisonnement peut être défini comme un enchaînement d'énoncés ou de représentations symboliques, conduit en fonction d'un but, ce but pouvant prendre des formes variées : démontrer, convaincre, élucider, interpréter, décider, justifier, expliquer, etc. Un tel enchaînement est en général non linéaire et nécessite des retours en arrière, présents dans la quasi-totalité des systèmes d'IA tout comme dans la démarche humaine.

Cette définition générale du raisonnement met en évidence l'étroite imbrication entre raisonnement et connaissances : aucun mécanisme ou modèle de raisonnement ne peut être séparé des connaissances sur lesquelles il opère.

Il existe une grande diversité de modes de raisonnement en IA, à l'image de la richesse des mécanismes mentaux de l'homme. La typologie ci-dessous correspond à des mécanismes de base, de nature différente mais non exclusifs :

- le raisonnement formel est fondé sur la manipulation syntaxique de structures symboliques à l'aide de règles, dans le cadre d'une sémantique. Le raisonnement logique en constitue évidemment l'archétype, mais il existe de nombreuses variantes et méthodes dérivées ;
- le raisonnement procédural dans lequel toutes les connaissances, la façon de les utiliser et la conduite du raisonnement sont entièrement figées sous forme d'algorithmes ;
- le raisonnement par analogie, très naturel et efficace chez l'être humain, mais difficile à mettre en œuvre en pratique car mal connu. Une variante simple, utilisée en pratique pour des applications de diagnostic ou de conception, est le raisonnement par cas. Le principe consiste à résoudre un problème nouveau en se fondant sur la solution d'un problème similaire, déjà rencontré, et rangé dans une mémoire de cas ;
- le raisonnement par généralisation et abstraction, également largement répandu chez l'homme, mais encore assez mal connu. Il est lié à l'apprentissage par induction. Ce raisonnement peut en particulier utiliser le mécanisme d'inférence par héritage et se rattache directement au raisonnement par classification.

Conserver la cohérence

Le problème central de la mécanisation d'un raisonnement est de parvenir à mener une suite d'opérations sur des structures symboliques, tout en préservant la véracité et la cohérence des conclusions déduites. La logique mathématique est un exemple parfait possédant cette propriété. Schématiquement, les raisonnements logiques peuvent être classés en trois grandes catégories :

- le raisonnement déductif permettant de déduire des conclusions valides à partir d'un ensemble de prémisses ;

- le raisonnement inductif visant, à l'inverse du précédent, à généraliser des prémisses ;
- le raisonnement abductif cherchant à attacher des causes plausibles à un ensemble de prémisses.

Des langages et des systèmes de représentation ont été construits sur la base du raisonnement déductif. Le plus connu est PROLOG, langage de programmation logique inventé à Marseille par A. Colmerauer et son équipe. PROLOG a été choisi par les Japonais comme langage de départ du projet d'ordinateurs de cinquième génération. Un autre exemple est SNARK développé par J. L. Laurière et son équipe. SNARK n'est pas un langage de programmation général comme PROLOG. Il se compose d'un langage de représentation des connaissances et d'un moteur d'inférence en logique des prédicats du premier ordre.

Le raisonnement logique présente des avantages importants pour la réalisation d'un système à bases de connaissances. Les règles de raisonnement sont syntaxiquement et sémantiquement définies ; elles sont de plus simples et faciles à mettre en œuvre. La clarté et la puissance d'expression de ce formalisme sont également séduisantes. Cependant, la logique possède de gros inconvénients dès que l'on aborde des domaines complexes, notamment du fait du manque d'efficacité des mécanismes de preuve.

Par ailleurs, la logique classique présente des limitations intrinsèques qui ne lui permettent pas d'aborder efficacement des problèmes de grande importance pratique tels que le traitement de l'incertain, la prise en compte du temps, la remise en cause de conclusions. Diverses extensions ont été proposées pour intégrer ces différents points.

La prise en compte de l'approximatif et du temps

La conception de systèmes à bases de connaissances performants nécessite donc de disposer de mécanismes de raisonnement approximatif efficaces, capables de prendre en compte les imperfections inhérentes aux connaissances humaines et aux données réelles, issues par exemple de capteurs physiques. Concevoir de tels mécanismes oblige à sortir du cadre strict de la logique mathématique classique.

Une première approche consiste à étendre la logique binaire traditionnelle en introduisant plusieurs valeurs de vérité entre le vrai et le faux. Les logiques multivaluées ont été largement étudiées sur le plan théorique. En revanche, leur utilisation pratique est encore limitée.

La solution adoptée dans la quasi-totalité des systèmes fait appel à une représentation numérique de l'approximatif. Il ne s'agit bien entendu que d'un pis-aller, car une théorie symbolique de l'approximation reste à construire. Différents formalismes sont utilisés : théorie des probabilités, théorie des possibilités, ensembles flous, théorie des fonctions de croyance de Dempster-Shafer, etc. On les trouve dans des applications de diagnostic (en médecine, dans l'industrie, etc.), d'aide à la décision (placement bancaire, etc.) d'interprétation de signaux (biomédicaux, industriels, militaires).

Le temps joue un rôle important dans les raisonnements. Raisonner sur le temps permet d'assurer un ensemble de fonctions de grande importance dans une application réelle :
- rendre compte de liens temporels entre des événements (positions relatives, écarts, durées, etc.) ;
- raisonner sur des événements à venir ;
- maintenir un ensemble de faits et de connaissances temporaires, éventuellement évolutifs ;
- planifier des actions et gérer leurs effets futurs ;
- maintenir des objectifs contraints dans le temps.

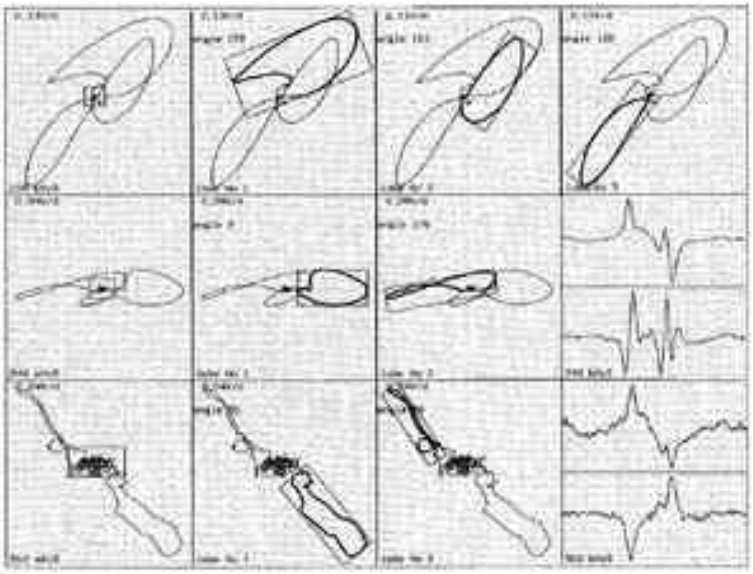
Il s'agit d'un domaine relativement récent mais très actif qui intervient dans de nombreux champs de l'IA : planification, compréhension du langage écrit ou parlé, interprétation de situations, conduite de procédés industriels, etc. Les diverses solutions proposées relèvent toutes d'extensions de la logique.

Un autre aspect du temps dans le raisonnement est relatif à la conception de systèmes capables de raisonner en temps réel. Il recouvre un ensemble de problèmes difficiles et mal résolus : interruption d'un raisonnement, prise en compte d'un flot de données provenant de capteurs, garantie d'un temps de réponse, etc.

Le rôle des modèles

Certains raisonnements, notamment dans le domaine du diagnostic de pannes, font appel à une modélisation du système étudié. Les modèles correspondent à une connaissance profonde des mécanismes sous-jacents qui vient compléter la connaissance heuristique exprimée sous forme de règles. Un expert humain se réfère souvent, devant un problème difficile, à un modèle mental de nature qualitative qu'il s'est forgé par expérience. Le raisonnement qualitatif mettant en œuvre de tels modèles joue ainsi un rôle important en IA.

La tendance actuelle est de mêler plusieurs types de connaissances et de raisonnements pour accroître l'efficacité des systèmes. Les progrès proviennent de nouvelles techniques de raisonnement (par exemple les systèmes experts de seconde génération s'appuyant sur des modèles, notamment liés à un raisonnement qualitatif) et de nouvelles architectures parallèles, plus efficaces.



La conjonction des techniques de traitement du signal, de reconnaissance de formes et de raisonnement hypothétique a permis la réalisation d'un système d'interprétation de signaux au sein du projet ESPRIT AITRAS auquel a participé le CRIN/INRIA Nancy. On voit ici l'utilisation de ce système pour la maintenance préventive de tubes d'échangeurs thermiques de centrale nucléaire par examen de signaux de courants de Foucault dans trois bandes de fréquence. Le raisonnement de l'ingénieur expert est de nature hypothético-déductive et se fonde notamment sur le contour et la forme des lobes des figures de Lissajous. Le système AITRAS reproduit cette démarche illustrée sur la figure.

Jean-Paul Haton, professeur à l'Université de Nancy I. Centre de recherche en informatique de Nancy (URA 262 CNRS), INRIA, BP 239, 54506 Vandœuvre-les-Nancy.

LES BASES DE CONNAISSANCES

Une base de connaissances n'est pas un logiciel. Elle regroupe sous une forme exploitable en informatique les connaissances du domaine que doit traiter un système expert.

KNOWLEDGE BASES - Establishing a knowledge base requires the input of an expert on the subject in question, a computer programmer to put it in a form that can be assimilated by a computer and a "cognician" who acts as a middleman. Once it is constituted, a knowledge base must be checked. Some formal methods allow to check inconsistencies of knowledge bases. However, heuristic approaches and testing are necessary for a more complete validation.

Marc Ayel
Marie-Christine Rousset

Il est nécessaire de distinguer deux niveaux bien distincts de connaissance. Le premier niveau constitue ce que l'on pourrait appeler "la théorie du domaine" et regroupe l'ensemble des connaissances de base : le vocabulaire fixe les noms des concepts ou objets de base du domaine et les noms de relations pouvant exister entre eux ; la sémantique est définie par l'expression des liens existant entre certaines relations, objets et concepts. Par exemple, dans un domaine médical très simple, la théorie du domaine pourrait contenir les éléments de connaissance suivants :

- parmi les maladies infectieuses, on distingue les bénignes des malignes,
- les maladies infectieuses peuvent être virales ou microbiennes,
- la grippe est une maladie infectieuse virale bénigne,
- la méningite est une maladie infectieuse maligne.

Le second niveau regroupe les connaissances de raisonnement d'un expert dans ce domaine : ces connaissances peuvent concerner la structuration de son raisonnement (par exemple : *commencer par prouver la présence d'une maladie de type infectieux avant d'essayer d'identifier une forme plus précise de maladie infectieuse*). Elles peuvent aussi se rapporter à la focalisation du raisonnement, de nature plus empirique (par exemple : *en cas de fièvre élevée, de maux de tête et de vomissements, penser à une méningite possible*).

Même si l'expertise d'un domaine n'a pas à être codée dans un langage informatique, elle doit être cependant suffisamment formalisée pour pouvoir être exploitée comme "données" par un moteur d'inférences adapté et constituer un système expert (voir encadré).

Les formalismes se fondent soit sur la logique, soit sur l'utilisation de règles de production, soit sur des représentations par objet. Compte tenu de la diversité des connaissances à représenter, on peut également utiliser des représentations hybrides associant règles et objets. Par exemple, une partie de la théorie du domaine, traduisant une hiérarchie de concepts, se prête naturellement à l'utilisation d'un formalisme par objet.

Une connaissance comme : *en cas de fièvre élevée, de maux de tête et de vomissements, penser à une méningite possible* pourra être traduite par une règle de production traduisant un lien d'évocation :
- fièvre-élevée et maux-de-tête et vomissements ⇒ diagnostic possible = méningite ;
ou bien encore, de façon différente, par une règle de production associée à un coefficient numérique dit de vraisemblance (ici 70 %) :
- fièvre-élevée et maux-de-tête et vomissements ⇒ diagnostic = méningite (0,7).

Ces différents formalismes ont en commun leur caractère déclaratif : ils sont constitués d'éléments de connaissances donnés en vrac, sans préjuger de l'ordre ou de la façon dont ils seront exploités par le moteur d'inférences.

Trois intervenants

La conception d'une base de connaissances fait appel à trois types d'intervenants :
- un expert du domaine d'application, puisque l'objectif est de se rapprocher au plus près de son expertise ;
- un informaticien, spécialiste des techniques d'intelligence artificielle, pour choisir le formalisme approprié de représentation des connaissances. Il devra aussi définir le moteur d'inférence le mieux adapté ;
- un "cogniticien", qui joue le rôle d'intermédiaire entre l'expert et l'informaticien, en aidant à l'explicitation des connaissances de l'expert sous une forme compréhensible aussi bien par l'expert que par l'informaticien.

La qualité finale du système expert dépend en grande partie de la qualité de la base de connaissances. Or, l'acquisition des connaissances auprès d'un expert n'est pas chose aisée car, précisément parce qu'il est expert, une grande partie des mécanismes qui sous-tendent son expertise lui sont devenus inconscients. Il est donc important de disposer de méthodologies et d'outils d'aide à l'acquisition et à la vérification des connaissances pour construire une base de connaissances fiable et reflétant le mieux possible l'expertise réelle.

En outre, les bases de connaissances ainsi constituées regroupent généralement une grande masse de connaissances en vrac : c'est le moteur d'inférences qui, pour chaque problème particulier, doit puiser dans la base de connaissances les éléments pertinents pour le problème à traiter et les organiser en vue de le résoudre. On a souvent intérêt à préparer le travail en compilant la base de connaissances sous une forme adaptée à un traitement efficace.

La compilation peut consister à indexer la base de connaissances en fonction des composants centraux pour une identification plus rapide des éléments de connaissances pertinents. Si la base de connaissances est un ensemble de règles, on peut par exemple indexer par mots-clés les attributs qui apparaissent dans la partie gauche des règles *si <...> alors <...>*. Une autre forme de compilation complémentaire consiste à construire un graphe de dépendances entre règles, permettant de mettre en évidence certaines séquentialités. Un tel ordre partiel pourra être exploité pour optimiser, en phase d'utilisation, la construction des enchaînements déductifs nécessaires à la résolution du problème courant.

Il existe actuellement deux manières d'aborder la tâche d'acquisition des

connaissances. La première relève des sciences cognitives et considère cette tâche comme une phase d'extraction des connaissances que l'expert a "dans la tête" en vue de reproduire fidèlement ses mécanismes de raisonnement.

La deuxième relève de la vision informatique et considère la phase d'acquisition comme une phase de modélisation, avec comme résultat un modèle conceptuel de l'application. Ce modèle est une formalisation intermédiaire, sorte de plate-forme commune, comprise par tous les intervenants chargés de l'application : l'expert du domaine, le cogniticien et l'informaticien. Pour construire le modèle conceptuel, certaines méthodologies se réfèrent à une bibliothèque de problèmes types et de tâches génériques déjà identifiées.

Supprimer les anomalies

La base de connaissances construite, il est essentiel de vérifier, au minimum, qu'elle ne contient pas d'anomalies, au mieux qu'elle correspond fidèlement à ce qu'elle est censée modéliser, c'est-à-dire l'expertise du domaine. La notion d'anomalie n'a pas de sens en soi et dépend du formalisme utilisé. Un certain nombre d'anomalies simples, relevant de l'erreur de syntaxe, sont faciles à détecter. D'autres proviennent des propriétés mêmes de la base de connaissances. Par exemple, si la base de connaissances est un ensemble de règles, on peut mettre en évidence des anomalies comme la présence :

- de règles inutiles, car redondantes avec d'autres ;
- de règles parasites, car jamais déclenchables ;
- de règles conflictuelles, car conduisant, dans des situations compatibles, à des conclusions contradictoires.

La deuxième phase de la vérification consiste à s'assurer que la base de connaissances correspond bien à ce qu'elle est censée modéliser. Sauf dans des cas très simples, il est impossible d'envisager une preuve automatique de correction et de complétude. On est donc contraint de faire appel à des études heuristiques.

Plutôt que d'étudier directement la base de connaissances, on peut aussi procéder par test sur différents cas représentatifs des problèmes auxquels le futur système expert sera confronté. L'objet du test n'est pas la base de connaissances seule mais l'ensemble "base de connaissances + moteur d'inférences", qui est observé en situation réelle et dont les résultats sont confrontés à ceux attendus par l'expert du domaine considéré. Celui-ci pourra alors repérer les incomplétudes et les distorsions éventuelles de la connaissance.

Dans cette présentation, nous avons traité la connaissance comme un matériau : il faut l'extraire, la vérifier, la reformuler... On peut pousser l'analogie un peu plus loin : la manipulation de la connaissance en tant que matériau exige des compétences et un savoir-faire spécifiques. Le but des recherches actuellement menées dans le domaine de l'acquisition et la vérification de la connaissance est précisément de fournir au "cogniticien" les moyens de mener à bien le développement d'un système manipulant de la connaissance. Ne peut-on pas alors parler à ce sujet de "génie de la connaissance" ?

SYSTÈMES EXPERTS : DE LA PREMIÈRE À LA SECONDE GÉNÉRATION

DENDRAL, MYCIN et PROSPECTOR ont marqué les débuts des systèmes experts au milieu des années soixante-dix. MYCIN, dû à Buchanan et Shortliffe, s'intéresse au diagnostic des infections bactériennes ; PROSPECTOR, développé au SRI par Hart et Duda, propose des sites de prospection minière et DENDRAL, réalisé par Buchanan, aide à la détermination de la structure de molécules en s'appuyant sur les résultats fournis par des spectromètres. Tous les trois furent suffisamment novateurs et eurent des résultats suffisamment probants pour susciter un intérêt très important non seulement de la part des chercheurs, mais aussi des industriels, ce qui déclencha une mode aux effets pas toujours favorables.

Une même structure informatique – base de connaissances, base de faits et moteur d'inférences – est caractéristique de ces systèmes experts de la première génération. Le moteur d'inférences étant indépendant du sujet traité, il peut être utilisé dans un autre domaine par substitution de la base de connaissances et de la base de faits. Ce concept a donné lieu à un développement commercial important et les sociétés de services ont vendu de nombreux générateurs, des plus simplistes aux plus sophistiqués. Certaines réalisations furent remarquables : le système R1 (XCON) est encore utilisé par les commerciaux de DEC pour définir les configurations d'ordinateur proposées à leurs clients.

Le passage à des réalisations opérationnelles et en vraie grandeur s'est cependant avéré difficile en raison des difficultés de modélisation d'un raisonnement d'expert, qui peut rarement se réduire à une suite de règles associatives. Il donna lieu au développement de systèmes dits systèmes experts de seconde génération, dans lesquels le modèle initial est complété par l'utilisation d'autres modèles (modèle profond, graphe causal) permettant par la coopération de différents points de vue et de différents types de connaissances, de se rapprocher plus finement du raisonnement sophistiqué développé spontanément par un expert humain face à des cas délicats.

Ce n'est donc pas un hasard si la modélisation du raisonnement du sens commun et celui des logiques non standards est en plein développement, si l'apprentissage et l'acquisition de connaissances se retrouvent au cœur de nombreux projets, si l'interaction homme-machine devient fondamentale avec la prise en compte de l'utilisateur. L'idée simple mais forte des systèmes experts en est à l'origine et leur mise en œuvre dans des applications opérationnelles va très certainement susciter de nouvelles questions et de nouveaux défis à la recherche en intelligence artificielle.

Marie-Odile Cordier, professeur à l'Université de Rennes I, IRISA (URA 227 CNRS), Campus universitaire de Beaulieu, 35042 Rennes Cedex.

Marc Ayel, professeur à l'Université de Savoie, directeur du Laboratoire d'intelligence artificielle, LIA - ESIGEC, Technolac, 73376 Le Bourget du Lac Cedex.

Marie-Christine Rousset, professeur à l'Université de Paris-Sud, Laboratoire de recherche en informatique (URA 410 CNRS), Université de Paris-Sud, Bât. 490, 91405 Orsay Cedex.

INFORMATIONS INCOMPLÈTES-INFORMATIONS CONTRADICTOIRES

L'homme sait raisonner lorsque l'information disponible est incomplète. Les conclusions obtenues résultent de connaissances tenues pour "généralement vraies". Ces conclusions sont provisoires car elles peuvent être remises en cause par l'arrivée de nouvelles informations qui viennent affiner ou contredire celles dont ont disposait auparavant.

INCOMPLETE INFORMATION - INCONSISTENT INFORMATION - Man is capable of reasoning when the information he has at his disposal is incomplete. The conclusions obtained in such a case are often only plausible, and should be defeasible when new information which contradicts them, becomes available. More than a dozen of non-classical formalisms dealing with uncertainty, either in a purely qualitative or in a quantitative manner, have been developed in artificial intelligence for rigorously modelling and implementing on computers this kind of reasoning.

Léa Sombé

Une grande part de nos connaissances est formée d'énoncés du type *les A sont B* (*Léa Sombé**), par exemple *les jeunes sont célibataires* ou *les oiseaux sont des êtres volants*. Ce type d'énoncé est souvent nuancé à l'aide d'expressions telles que "généralement", "sauf exception", "typiquement", "dans la plupart des cas", ..., ou peut même être réécrit sous la forme d'énoncés à caractère graduel tels que *plus on est jeune, plus on a des chances d'être célibataire*, si les propriétés considérées peuvent être satisfaites à divers degrés (comme ici *jeune*). Longtemps la logique classique a été unanimement considérée comme un modèle suffisant pour formaliser le raisonnement; mais la traduction en logique classique de *les A sont B* par *tous les A sont B* interdit la possibilité d'exception. On pourrait penser contourner la difficulté en écrivant *tous les A sauf les C sont B*, dont l'expression en logique suppose explicitement connus tous les C. Mais il n'est pas réaliste de prétendre pouvoir toujours exprimer toutes les exceptions. Par ailleurs, la connaissance de nouvelles exceptions obligera à reconsidérer totalement la représentation des connaissances. De plus, répondre B nécessite alors d'établir non seulement A mais encore de vérifier qu'on n'est dans aucun des cas d'exception; cela suppose que l'information nécessaire est disponible. Or on peut savoir qu'on a affaire à un A sans savoir pour autant si c'est ou non un C.

La logique classique n'offre pas de mécanisme pour la remise en cause de conclusions, à l'arrivée de nouvelles informations. Ceci est dû essentiellement à une propriété de monotonie (voir encadré "déduction non-monotone"). Pour les formalismes non-classiques développés en intelligence artificielle, l'énoncé *les A sont B* est entendu sous une forme affaiblie du type *généralement les A sont B*. Chaque formalisme décrit le *généralement* à sa manière. Les "traductions" possibles sont variées et suggèrent différents traitements de ces énoncés : *un A est un B, sauf si B est connu comme faux pour cet A* (logique des défauts), *si un A donné n'était pas un B, on le saurait* (logique autoépistémique), *tout A qui n'est pas anormal est un B* (circonscription), *les A typiques sont des B* (logique des conditionnels), *la plupart des A sont des B* (logique probabiliste), *un A est presque certainement un B* (logique possibiliste), pour ne faire référence qu'à quelques-unes des approches les plus développées. Se pose alors le problème de redéfinir ce qu'on entend par "inférence" à partir de tels énoncés, et de préciser les propriétés attendues du mécanisme de déduction associé. Il n'est pas possible de rentrer ici dans les aspects techniques de chaque formalisme.

Dans le raisonnement plausible, le manque d'information ne permet quelquefois que de dériver des conclusions provisoires et fragiles. Ceci conduit à des situations où ces conclusions rentrent en contradiction avec de nouvelles informations (et doivent donc être modifiées). La problématique de la révision des bases de connaissances (représentées dans un formalisme logique), ne se place pas au niveau de l'inférence mais plutôt au niveau du maintien de leur cohérence en cas d'ajout d'information.

Deux raisons de réviser une base

De façon générale, un processus de révision d'une base de connaissances $\mathcal{K}$ consiste à modifier cette base $\mathcal{K}$ en une nouvelle base $\mathcal{K}'$ pour prendre en compte l'arrivée d'un ensemble de nouvelles informations. Il semble qu'un tel processus puisse être envisagé dans deux types de situations différentes conduisant ainsi à deux problématiques quelque peu distinctes : l'évaluation des croyances d'un agent (ou d'un système informatique) d'une part, et la description des changements intervenant dans un monde évolutif, d'autre part.

Dans le premier cas, le système maintient des croyances sur la vérité ou la fausseté de propositions sur lesquelles l'information manque au moins partiellement ou est empreinte d'incertitude. L'arrivée de nouvelles informations, dans la mesure où elles vont à l'encontre de ce qui était crédible, oblige à une révision des croyances courantes maintenues par le système. Dans ce cas, l'idée d'ordre sur les croyances (éventuellement exprimé numériquement) apparaît naturellement, certaines choses pouvant être davantage crédibles que d'autres. Cet ordre permet de guider la procédure de restauration de la cohérence nécessitée par l'arrivée de nouvelles informations, en éliminant les énoncés les moins crédibles qui créent cette incohérence. Dans cette optique, le processus de la révision cherche à maintenir ou à augmenter la qualité des argu-

ments qui pourraient être fournis à l'appui des croyances du système. La révision de croyances suite à des interactions entre agents, ou le diagnostic de pannes en environnement non-évolutif (toutes les informations sur le système défaillant n'étant pas disponibles au même moment ou au même coût) offrent des exemples de ce type de processus.

Dans la seconde situation, la description d'un monde sur lequel on dispose en général d'une information relativement complète, doit être mise à jour à l'arrivée d'informations vraies sur le monde, mais contradictoires avec la description que l'on en avait. Ces nouvelles informations peuvent décrire les conséquences d'une action effectuée ou refléter simplement une observation faite sur le monde qui a évolué. Ce qui importe alors est d'être fidèle à l'évolution du monde. L'aspect chronologique du problème est clairement important dans ce suivi de l'évolution du monde. Par ailleurs, la nécessité d'un retour à une consistance forte dans le passage de K à K' est manifeste dans cette problématique puisque la description du monde doit rester cohérente, alors qu'il est éventuellement plus acceptable d'avoir des incohérences partielles ou "locales" dans un état de croyance (des logiques dites "para-consistantes" sont d'ailleurs développées pour supporter de telles incohérences locales).

Ce que l'on croit du monde et ce que l'on en sait

Préserver la cohérence ne suffit pas: en général, on souhaite que le processus de révision obéisse de plus à d'autres critères : en particulier que priorité soit donnée à la nouvelle information, et que le changement opéré dans le passage de K à K' soit "minimal" dans un certain sens. La priorité à la nouvelle information ne va pas de soi si cette information s'avère plus imprécise (ou incertaine) que ce qui était connu. De ce point de vue, la révision se différencie de la "fusion" d'informations provenant de sources distinctes, où les différentes sources sont considérées sur un même pied, même si l'on prend en compte la confiance relative que l'on a dans chacune d'elles. La minimalité du changement peut s'exprimer de diverses façons ; on cherche par exemple à remettre en cause le moins de croyances ou les croyances les moins avérées, à minimiser une mesure scalaire d'information (en particulier dans les modèles probabilistes). La formalisation de mécanismes de révision offrant des garanties théoriques et répondant aux besoins des applications, soulève donc des questions délicates (voir encadré "révision") dont beaucoup n'ont pas encore trouvé de réponses définitives.

RÉVISION "SYNTAXIQUE" OU "SÉMANTIQUE" ?

L'opération de révision apparaît différente selon qu'elle est faite sur un ensemble d'énoncés exprimés en logique classique, éventuellement entachés d'incertitude) ou sur l'ensemble des interprétations du monde (qui peuvent être pondérées par une mesure d'incertitude) compatibles avec la base de connaissances.

Considérons l'exemple très simple suivant. On a deux bases de connaissances K_1 et K_2, contenant chacune deux énoncés (la flèche $\rightarrow$ se lit "si ... alors ...")

$K_1 = (a, a \rightarrow b)$ $\qquad$ $K_2 = (a, b)$

Ces deux bases syntaxiquement distinctes ont le même contenu sémantique : de chacune d'elles on peut déduire toutes les conséquences de la conjonction logique $a \wedge b$. En d'autres termes, sur les quatre interprétations, c'est-à-dire les quatre situations possibles exprimables en termes de a et de b, à savoir : "a et b vrais", "a et $\neg$ b vrais", "$\neg$ a et b vrais", "$\neg$ a et $\neg$ b vrais", seule "a et b vrais" est compatible avec K_1 ($\neg$ dénote la négation). Cette interprétation est aussi la seule possible pour K_2. Imaginons maintenant l'arrivée de l'information "$\neg$ b". On peut alors avoir envie de réviser K_1 et K_2 différemment, bien qu'elles aient le même contenu sémantique : par exemple on peut modifier K_1 en $K'_1 = (\neg b, a \rightarrow b)$ et K_2 en $K'_2 = (a, \neg b)$ qui n'ont pas le même ensemble de conséquences (K'_1 permet de déduire $\neg a$).

Bien entendu, si on ne considérait que le contenu sémantique, les révisions de K_1 et K_2 seraient identiques. Les approches numériques les plus développées privilégient généralement la représentation sémantique alors pondérée par une mesure d'incertitude : par exemple, dans l'approche bayésienne, une distribution de probabilité sur les interprétations est modifiée en présence d'une observation certaine (ou incertaine), en respectant un principe de minimum d'entropie relative. Il est cependant clair que les différences "syntaxiques" (par exemple entre K_1 et K_2) sont importantes à prendre en compte, parce que la forme sous laquelle l'information est donnée véhicule très souvent des renseignements sur la provenance, l'importance ou l'intérêt de tel ou tel énoncé.

DÉDUCTION NON-MONOTONE

La logique classique peut être caractérisée en terme d'un opérateur de déduction, noté $\vdash$, et reliant des formules où :

$A \vdash B$ signifie : "de A on peut dériver B". Cet opérateur vérifie les propriétés suivantes :

$A \vdash A$ (réflexivité)

si $A \vdash B$, $B \vdash C$, alors $A \vdash C$ (transitivité)

si $A \vdash B$, alors $A \wedge A' \vdash B$ (monotonie)

En logique classique, lorsqu'on augmente la base de connaissances avec des nouvelles informations, l'ensemble des conséquences qui peuvent en être déduites ne peut qu'augmenter (c'est-à-dire qu'il croît de façon monotone au sens de l'inclusion). Les logiques qui cherchent à formaliser le raisonnement plausible rompent avec la propriété de monotonie de l'opérateur de déduction, ainsi qu'avec la propriété de transitivité qui est liée à la précédente. L'abandon de la propriété de monotonie est motivée par des exemples de raisonnements tels que celui-ci : on sait que généralement les oiseaux volent, donc si on sait que Tin est un oiseau, alors on en conclura plausiblement qu'il vole, ce qui peut s'écrire : oiseau (Titi) $\vdash$ vole (Titi), en revanche bien que les pingouins soient des oiseaux, on ne souhaite pas avoir : [oiseaux (Titi), pingouin (Titi)] $\vdash$ [vole (Titi)], ce que nous impose la propriété de monotonie de $\vdash$.

La déduction non-monotone, notée $\mid\sim$, affaiblit les propriétés de monotonie et de transitivité, par exemple sous la forme suivante :

si $A \mid\sim B$ et $A \mid\sim C$, alors $A \wedge B \mid\sim C$;

si $A \mid\sim B$ et $A \wedge B \mid\sim C$, alors $A \mid\sim C$.

* *Léa Sombé* est un groupe composé actuellement de onze personnes appartenant à des laboratoires participant au Groupement de recherche/Programme de recherches coordonnées "Intelligence artificielle" :
CERT-ONERA (Toulouse) : L. Cholvy - GIA URA 816 CNRS (Marseille) : C. Schwind, P. Siegel - INRIA URA 227 CNRS (Rennes) : Ph. Besnard, M. O. Cordier, Y. Moinard - IRIT URA 1399 CNRS (Toulouse) : D. Dubois, L. Farinas del Cerro, H. Prade - LIPN (Université de Paris-Nord) : F. Lévy - LRI URA 410 CNRS (Université de Paris-Sud) : C. Froidevaux.

L'APPRENTISSAGE

En intelligence artificielle, le processus d'apprentissage peut être centré ou non sur l'acquisition de connaissances à partir d'un expert. Pour ceci, il peut utiliser des méthodes déductives, inductives ou analogiques. Il utilise non seulement des connaissances, mais aussi des métaconnaissances qui aident à comprendre les raisons des succès et des échecs.

KNOWLEDGE ACQUISITION - Acquisition of knowledge by computers may be performed directly by the machine or through the mediation of human beings. The process used may be deduction, induction or analogy. An intelligent system may even make use of meta-knowledge, i.e. knowledge about knowledge.

Yves Kodratoff

Le mot "apprentissage" a trois sens en intelligence artificielle.

Le premier correspond à l'acquisition de connaissances par l'homme. Il concerne l'enseignement assisté par ordinateur qui s'occupe de la création et de l'utilisation de modèles cognitifs d'un apprenant humain, dans le but de l'aider à apprendre.

Dans le deuxième sens, un homme aide une machine à acquérir des connaissances. Ce domaine est centré sur les interfaces conviviales permettant à l'expert d'exprimer sa connaissance dans un formalisme compréhensible par la machine. Durant les cinq dernières années, le sens du terme *knowledge acquisition* a évolué. On demande aujourd'hui aux interfaces conviviales des capacités de raisonnement inductif ou déductif qui leur permettent de fournir une aide raisonnée à leur utilisateur.

Le troisième sens concerne l'acquisition de connaissances par la machine elle-même. On part d'un ensemble de connaissances conduisant à un premier comportement. On se donne une fonction d'évaluation des résultats obtenus, et on corrige la connaissance en fonction de la qualité des résultats. On conserve ainsi progressivement la seule connaissance qui donne de bons résultats.

Parallèlement sont nées, durant la dernière décennie, des méthodes qui tentent de formaliser le phénomène d'extraction de connaissance à partir de données de natures diverses. Les données peuvent être des exemples de concepts ou de comportement, elles peuvent être accompagnées ou non de connaissances supplémentaires explicites. Les raisonnements effectués peuvent être de nature déductive, inductive, ou analogique (voir figure).

Le raisonnement déductif est toujours utile

L'importance apportée au raisonnement déductif pour l'apprentissage étonne toujours un peu les non-spécialistes. De fait, apprendre des connaissances nouvelles ne se limite pas à l'acquisition de nouveaux concepts ou de nouvelles relations entre les concepts. L'apprentissage comprend aussi l'acquisition de nouvelles stratégies pour mieux utiliser les "vieilles" connaissances, et ceci peut se faire par un raisonnement déductif.

Apprendre à partir de spécifications consiste à transformer des relations en procédures. La relation est par exemple le commandement du code de la route : *il faut se tenir sur la partie droite de la chaussée*, et la procédure l'ensemble des comportements qui permettent de rester effectivement sur la partie droite de la chaussée.

L'apprentissage déductif à partir d'exemples s'impose quand, malgré une connaissance théorique parfaite du domaine, les procédures de résolution du problème s'avèrent impraticables. On analyse un exemple de solution (élégante!) et on extrait une stratégie pour résoudre de nouveaux problèmes avec élégance. Ce thème de recherche a connu un développement rapide dans les cinq dernières années.

L'histoire de ces dix dernières années a été faite (et celle des dix prochaines le sera aussi) des efforts pour passer d'un apprentissage inductif empirique, qui n'utilise pas les connaissances du domaine, à un apprentissage inductif constructif qui, au contraire, s'appuie fortement sur les connaissances déjà acquises avant que

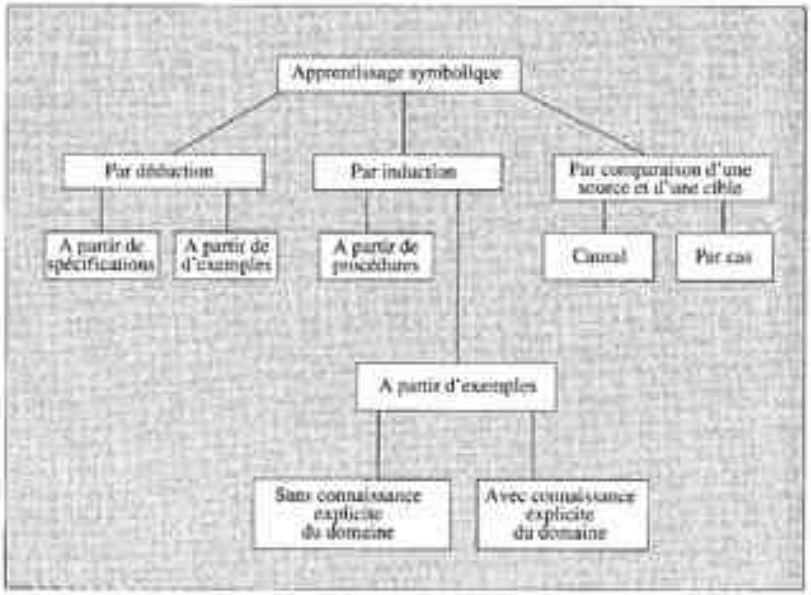


Tableau simplifié des techniques de raisonnement utilisées pour effectuer l'apprentissage.

l'apprentissage ne commence. Un tel apprentissage est aujourd'hui qualifié de "multi-stratégique".

Ressemblances et différences

Le raisonnement par cas et le raisonnement causal détectent des ressemblances et des différences appelées relations de similarité. Roméo disant à Juliette qu'elle est "son soleil", détecte une similarité entre le soleil et Juliette, en l'occurrence, que le soleil est un objet astronomique qui conditionne la vie sur les planètes qu'il attire, et que Juliette est un objet humain qui conditionne la vie de Roméo qu'elle attire. Les propriétés physiques du soleil correspondent également à des propriétés psychologiques de Juliette.

Le raisonnement par cas, qui commence à concurrencer le raisonnement par règles (celui effectué par les systèmes experts), est très semblable au raisonnement analogique, mais il ne recherche pas des relations de causalité. Il recherche des relations de proximité. Le concept de Juliette est très éloigné du concept de soleil, et on ne peut comparer leurs cas. Mais Juliette est très proche de Marguerite, dans Faust, (deux jeunes filles innocentes etc.) et on peut essayer d'appliquer le cas de Juliette à celui de Marguerite.

Les processus d'apprentissage sont aujourd'hui utilisés dans les domaines les plus variés : de la gestion des prêts bancaires à la détection des défauts des pales d'hélicoptères, en passant par la sélection des pilotes de l'US Air-Force. De nombreuses comparaisons ont montré qu'un système obtenu par interrogation directe d'un expert est systématiquement moins efficace (plus grand nombre d'erreurs) qu'un système dont les règles ont été obtenues par apprentissage. Par ailleurs, le nombre des règles utilisées dans un système construit par apprentissage est très inférieur à celui obtenu par interrogation directe de l'expert pour des résultats comparables.

Yves Kodratoff, directeur de recherche au CNRS, Laboratoire de recherche en informatique (URA 410 CNRS), Université de Paris-Sud, Bât. 490, 91405 Orsay Cedex.

LES MÉTACONNAISSANCES

Un système d'intelligence artificielle utilise des connaissances sur le domaine où il travaille, par exemple sur les mathématiques ou la médecine. Mais quand il raisonne sur ce qu'il fait, il lui faut des "connaissances sur les connaissances" qui sont appelées métaconnaissances. Nous en utilisons nous-mêmes constamment sans nous en rendre compte, et il existe plusieurs variétés de métaconnaissances. Certaines sont des caractéristiques des connaissances, comme leur importance ou leur certitude : dans *Charles VII était peut-être le fils de Charles VI*, nous avons d'une part une connaissance historique et d'autre part une métaconnaissance qui est un doute sur la véracité de la connaissance précédente ; elle est indiquée par le mot "peut-être". *Jean croit au Père Noël* est aussi une métaconnaissance : c'est une connaissance sur les connaissances de Jean. D'autres métaconnaissances agissent sur les connaissances, par exemple pour en inventer de nouvelles. *Si un concept a deux caractéristiques de même nature, envisager le cas où elles ont la même valeur* amène, pour la multiplication, à considérer le cas où les deux opérandes sont les mêmes, donc à découvrir le concept de "carré". Mais cette métaconnaissance est valable dans d'autres domaines que les mathématiques. En s'inspirant d'elle, certains auteurs de romans policiers fusionnent deux des principaux rôles : la victime, le coupable et le policier. Ainsi, la fusion des rôles de victime et de policier amène un effet comique alors que celle de coupable et de policier déroute le lecteur comme dans quelques uns des meilleurs romans d'Agatha Christie.

A quoi servent les métaconnaissances ? Un système intelligent doit pouvoir manipuler les connaissances dont il dispose afin d'expliquer ce qu'il a trouvé, d'apprendre en comprenant les raisons de ses succès et de ses échecs. Cela demande de travailler à deux niveaux : au niveau de base, on résout le problème et au niveau "méta" supérieur, on observe ce que l'on fait pendant que l'on résout le problème ; cette observation permet de donner les explications demandées et d'analyser ce que l'on a fait bien ou mal. De même, il n'est pas bon de résoudre un problème en faisant uniquement des essais parmi ce qui est autorisé, il faut aussi manipuler l'énoncé de ce problème. S'il s'agit de trouver deux entiers m et n (qui peuvent être négatifs) satisfaisant $4m+3n^2=34$, on peut essayer tous les couples de valeurs de m et de n jusqu'à ce que l'on en trouve satisfaisant cette équation. Mais un système peut aussi raisonner sur l'énoncé du problème : 34 et $4m$ sont pairs, donc leur différence $3n^2$ est aussi paire, donc n^2 est pair, donc n est pair. Mais alors n^2 est multiple de 4 et, par conséquent, $4m+3n^2$ est également multiple de 4. Or le second membre, 34, n'est pas multiple de 4, donc le problème n'a pas de solution. En travaillant sur l'énoncé, donc à un niveau méta, on gagne ici un temps considérable par rapport au programme qui énumère sans espoir tous les couples de valeurs possibles pour m et n. L'énoncé du problème est aussi un objet d'étude, étude qui se fait à l'aide de métaconnaissances.

Nous nous servons constamment de métaconnaissances : nous ne naissons pas avec la connaissance d'une langue, mais avec des métaconnaissances pour apprendre une langue ; grâce à elles, nous apprenons la langue du milieu dans lequel nous vivons. Un proverbe africain dit : *si vous donnez un poisson à un homme, il n'aura pas faim pendant un jour, mais si vous lui apprenez à pêcher, il n'aura plus faim pour le reste de sa vie*. Donner des connaissances à un système est analogue à donner un poisson, lui donner des métaconnaissances est analogue à apprendre à pêcher.

Jacques Pitrat, directeur de recherche au CNRS, LAFORIA (URA 1095 CNRS), Université Pierre et Marie Curie, Tour 46-00, 4, place Jussieu, 75252 Paris Cedex 05.

LES RÉSEAUX DE NEURONES FORMELS

Les réseaux de neurones formels peuvent réaliser une approximation de n'importe quelle relation non linéaire, ce qui leur ouvre des domaines très divers : la reconnaissance des formes, la modélisation et la commande de systèmes non linéaires, etc. Il s'agit là d'une technique très différente, mais parfois complémentaire de l'intelligence artificielle.

FORMAL NEURAL NETWORKS - Neural networks are universal approximators: they can appropriate any non linear input-output mapping. This striking property opens up various fields of application, such as automatic classification (for pattern recognition and expertise gathering through examples), industrial process modeling, non-linear system control, etc. Although this approach is very different from artificial intelligence, neural networks may sometimes act as substitutes to symbolic processing systems.

Gérard Dreyfus

Les recherches sur les réseaux de neurones formels ont été stimulées par deux facteurs :
- l'enjeu industriel de tâches complexes telles que la reconnaissance des formes, le traitement de signaux temporels, la commande de processus ; en effet, il est tentant de rechercher dans la structure des systèmes nerveux une inspiration pour concevoir des machines susceptibles de remplir des fonctions que les systèmes actuels de traitement de l'information réalisent malaisément ;
- la nécessité d'apporter à la modélisation en neurobiologie ou neuropsychologie des concepts nouveaux permettant éventuellement de faciliter l'interprétation des données expérimentales disponibles et, mieux encore, de suggérer de nouvelles expériences et d'en prédire les résultats.

De nouvelles perspectives ont été ouvertes depuis 1982, grâce à deux contributions significatives :
- l'application des concepts issus de la physique des systèmes désordonnés a permis d'analyser et de prévoir quantitativement le comportement de réseaux de neurones formels construits selon certaines architectures ;
- l'introduction de nouvelles méthodes d'apprentissage a autorisé la conception de réseaux de neurones qui vont au-delà des limitations des réseaux connus antérieurement.

Ces deux contributions sont de nature très différente : la première est de nature fondamentale et constitue une avancée conceptuelle majeure ; la seconde est de nature algorithmique : elle a eu des retombées dans le domaine des applications industrielles.

Une lointaine ressemblance avec les neurones biologiques

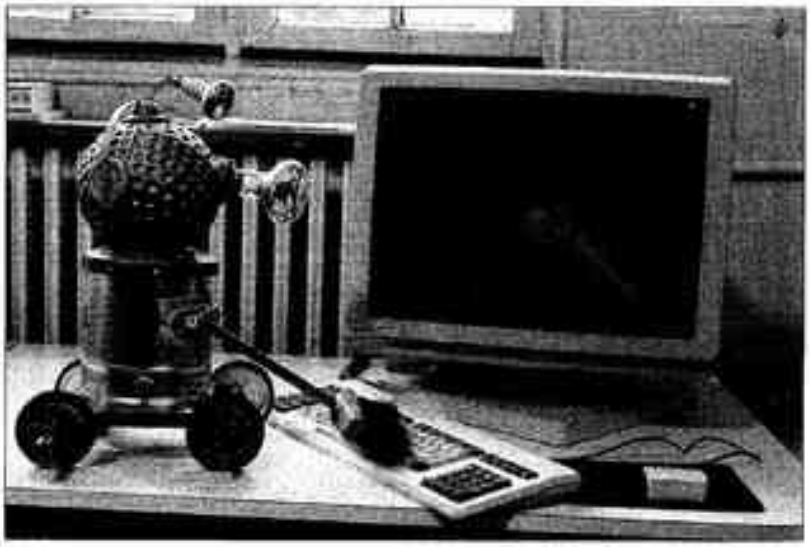
Le terme de "réseaux de neurones formels" est justifié par le fait que les systèmes "neuronaux" artificiels possèdent quelques-unes des caractéristiques des réseaux de neurones vivants : ils sont constitués de processeurs nombreux, non linéaires, fortement connectés entre eux, fonctionnant de manière parallèle et coopérative, dans une structure qui peut être soit figée, soit en adaptation permanente. Ils peuvent réaliser une grande variété de fonctions. Néanmoins, le nombre réduit de processeurs mis en jeu, la simplicité du neurone élémentaire, l'insuffisance de nos outils conceptuels, font que les réseaux de neurones formels sont encore très loin d'atteindre les performances des réseaux de neurones des animaux les moins évolués.

L'élément de base des réseaux de neurones formels est un processeur très simple, qui effectue une somme pondérée de ses entrées (lesquelles peuvent être des entrées du réseau, ou les sorties d'autres neurones du réseau) et qui calcule une fonction non linéaire de cette somme. Les coefficients de pondération sont appelés "coefficients synaptiques". La fonction non linéaire est souvent une fonction non linéaire monotone croissante entre -1 et +1.

Un réseau de tels neurones interconnectés est un système de calcul parallèle puisque tous les neurones prennent leurs décisions simultanément.

Ils sont capables d'apprentissage

L'apprentissage consiste à déterminer les coefficients d'interaction entre les neurones. On en distingue deux types :
- l'apprentissage supervisé, au cours duquel les coefficients sont calculés de telle manière que le réseau présente un comportement aussi voisin que possible d'un comportement désiré, déterminé par un "professeur" ;



Réseaux de neurones et robotique font bon ménage (Cliché ESCPI - D. Morisseau).

- l'apprentissage non supervisé, au cours duquel le réseau est censé s'organiser par lui-même, sans intervention d'un professeur.

Nous ne considérerons ici que l'apprentissage supervisé. Un tel apprentissage peut s'effectuer de manière adaptative ou non adaptative :

- pour les systèmes à apprentissage non adaptatif, on distingue une phase initiale d'apprentissage, au cours de laquelle les coefficients synaptiques sont calculés, suivie d'une phase au cours de laquelle le réseau est utilisé avec les coefficients calculés pendant l'apprentissage ;
- pour les systèmes à apprentissage adaptatif, les phases d'apprentissage et d'utilisation sont confondues : le réseau apprend en permanence.

Les fonctions réalisées par les réseaux de neurones formels sont directement liées à leur architecture : les réseaux non bouclés sont utilisés essentiellement pour remplir des tâches de classification, tandis que les réseaux de neurones bouclés, au comportement plus complexe, interviennent dans le domaine du filtrage adaptatif et de la commande de processus, ainsi que sous la forme de mémoires associatives.

Réseaux non bouclés ...

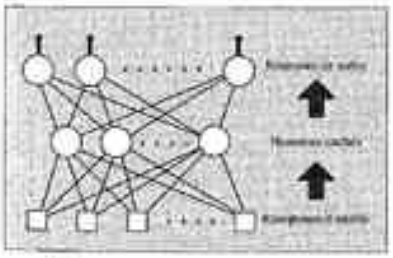
Les réseaux non bouclés sont des réseaux dont le fonctionnement ne fait pas intervenir la notion de temps : le calcul de la somme pondérée et celui de la fonction non linéaire se font de manière quasi-instantanée. La figure ci-dessous représente un exemple de structure de réseau non bouclé, dite "structure en couche", dans lequel l'information passe des récepteurs d'entrée à une couche de neurones intermédiaires dits "neurones cachés", puis à une couche de neurones de sortie. La présence des "neurones cachés" n'est pas indispensable : leur utilité dépend du problème de classification considéré.

Supposons que nous désirions reconnaître des chiffres manuscrits (de 0 à 9), constitués d'un nombre déterminé de points. Nous avons besoin d'autant de récepteurs qu'il y a de points dans l'image, et de dix neurones de sortie. Au cours de l'apprentissage, les coefficients d'interaction entre les neurones sont calculés de telle manière que chaque neurone de sortie se "spécialise" dans la reconnaissance d'un des dix chiffres de 0 à 9 : par exemple, le neurone n° 0 apprend à répondre +1 si l'exemple présenté au récepteur d'entrée est un zéro, et à répondre -1 si c'est un autre chiffre. Un chiffre est donc reconnu si un seul des dix neurones répond +1 et tous les autres -1, sinon la forme est considérée comme ambiguë ou non reconnaissable.

Le réseau ne doit pas seulement être capable "d'apprendre par cœur" tout l'ensemble d'apprentissage ; il doit aussi être capable de classer correctement des formes inconnues. Prédire les capacités de généralisation d'un réseau d'architecture donnée et, mieux encore, trouver la structure de réseau ayant les meilleures performances de généralisation pour un problème donné, tels sont les objectifs de nombreuses équipes de recherche.

Dans l'état actuel des connaissances, on ne connaît pas de raison fondamentale pour que les réseaux de neurones utilisés pour la classification (en vision artificielle, reconnaissance de la parole, contrôle non destructif, etc.) aient des performances meilleures que les classifieurs "non neuronaux" de complexité équivalente ; en revanche, ils peuvent avoir un avantage sur ces derniers en ce qui concerne la vitesse de reconnaissance, dans la mesure où ils sont implantés sur des structures parallèles (ordinateurs spécialisés ou circuits intégrés spécifiques).

Les réseaux non bouclés peuvent également être utilisés pour effectuer des tâches de filtrage non linéaire, de modélisation de systèmes non linéaires, et de commande de processus ; dans ces applications, il est souvent utile de recourir à un apprentissage adaptatif.



Un réseau de neurones non bouclé.

... ou bouclés

Les réseaux de neurones bouclés pour mémoire associative mettent à profit l'évolution spontanée d'un réseau de neurones bouclé vers un état "attracteur", en l'absence de signaux d'entrée ; ils ont été étudiés en très grand détail, mais leurs applications pratiques sont restées très limitées. En revanche, les réseaux de neurones dynamiques fonctionnant en régime forcé, c'est-à-dire en présence de signaux d'entrée en évolution permanente, présentent un grand intérêt pratique ; ils constituent en effet des filtres récursifs non linéaires qui peuvent servir en filtrage adaptatif, notamment pour la transmission d'information (compression de la parole, égalisation de canaux), ou encore pour construire des modèles de processus dynamiques non linéaires. De plus, les réseaux de neurones bouclés permettent de constituer des systèmes de commande, éventuellement adaptatifs, de processus non linéaires. Dans ce domaine, les réseaux de neurones n'apportent pas seulement des avantages en termes de vitesse d'exécution, mais ils permettent surtout de réaliser des fonctions qui n'étaient pas réalisables (ou réalisables de manière très compliquée) par d'autres méthodes. Des réseaux de neurones pour la modélisation de systèmes complexes ou pour la commande de processus non linéaires sont d'ores et déjà opérationnels en milieu industriel. Il s'agit donc là d'un domaine particulièrement prometteur.

NEURONES EN SILICIUM

La plupart des études sur les réseaux de neurones ont été faites jusqu'ici par simulation sur ordinateur. Mais il commence à apparaître des circuits intégrés spécifiques qui rassemblent sur une "puce" un réseau neuronal utilisable.

Actuellement, un seul circuit intégré neuronal est commercialisé – par Intel – à un prix d'ailleurs prohibitif (plusieurs milliers de francs), pour des applications courantes. Il rassemble soit une couche de 64 neurones tous connectés, ou deux couches de 64 neurones non complètement connectés. Il fonctionne d'une manière analogique, les connexions étant codées sur 4 bits.

Plusieurs laboratoires ont produit des circuits expérimentaux. Carver Mead du California Institute of Technology a mis au point une rétine artificielle permettant, avec une centaine de neurones analogiques, la reconnaissance d'un contour simple ou la détection de mouvements. Une variante permet de reconnaître les sons (cochlée artificielle). La société AT&T a mis au point un réseau de reconnaissance de chiffres manuscrits, pour la lecture des codes postaux, avec un taux de reconnaissance exacte de 90 %, 1 % d'erreurs et 9 % de chiffres non classés.

Dans le domaine numérique, Hitachi a présenté en 1989 un énorme circuit de 180 neurones tous connectés, dont l'état est défini sur 9 bits et les coefficients synaptiques sur 8 bits.

Enfin, en France, plusieurs circuits intégrés ont été conçus et réalisés, tant dans des laboratoires industriels qu'universitaires.

Gérard Dreyfus, professeur à l'Ecole supérieure de physique et de chimie industrielles de la ville de Paris (ESPCI), directeur du Laboratoire d'électronique, 10, rue Vauquelin, 75005 Paris.

DE L'INFORMATIQUE À LA COGNITION

Comme science du calcul effectif, l'informatique a permis de donner une définition scientifique de la cognition.

FROM INFORMATION TECHNOLOGY TO COGNITION - A parallel is drawn between mental processes and the operation of computers. The neural approach, which goes as far as assimilating the structures of biological neurons to electronic neural systems, is particularly productive. However, whether or not this actually reflects the facts remains to be established.

Mario Borillo

L'objet des sciences de la cognition est d'abord l'étude des aptitudes et fonctions essentielles de l'esprit humain, comme le langage, l'apprentissage, le raisonnement, l'intention, l'action planifiée... Le langage, par exemple, ne se réduit pas à une famille de structures ondulatoires associées au signal de parole, et l'action planifiée à une distribution spatiotemporelle de modifications provoquées de l'univers physique. C'est par les significations associées au message linguistique dans le premier cas, les intentions et les stratégies du planificateur dans le second, que se traduit la dimension cognitive. Or, ni les unes ni les autres ne sont observables en elles-mêmes. Il est donc nécessaire de définir un cadre théorique à l'intérieur duquel elles puissent être définies avec précision et articulées dans une architecture conceptuelle cohérente, ainsi qu'une méthodologie permettant de lier des hypothèses relatives à ces entités empiriquement inaccessibles avec des observations systématiques sur les comportements (ici, le langage ou l'action) par lesquels elles se manifestent.

Deux principes de base

L'informatique a apporté ce paradigme fondateur sans lequel, littéralement, les sciences cognitives n'auraient pu être conçues. Leur édifice repose sur deux principes de base, fortement réducteurs peut-être, mais qui permettent de progresser. Selon le premier, "les processus mentaux peuvent être vus comme des fonctions admettant les stimuli comme arguments, au même titre que les programmes sont définis comme des fonctions permettant de passer des données au résultat de l'exécution". Le second reprend la vieille intuition de Hobbes sur le raisonnement comme par calcul et l'actualise à la lumière de la théorie des automates et de la thèse de Church, en posant que la théorie de la calculabilité effective circonscrit les limites de la pensée. A partir de ces fondations, le parallèle avec l'informatique est développé systématiquement pour rendre compte des questions les plus importantes. Ainsi, la relation entre les processus mentaux (la pensée) et le cerveau est conçue comme étant du même ordre que celle qui existe entre les composantes logicielle et matérielle d'un système informatique. Plus précisément encore, le système cognitif est décrit en termes d'états internes et de processus par lesquels le système passe d'un état à l'autre, en mobilisant divers types de mémoires et de processeurs. L'isomorphisme fonctionnel entre processus mentaux et fonctionnement de l'automate a été poussé assez loin pour qu'on ait pu écrire : "Le langage naturel est un langage de programmation effectuant les transitions cognitives entre les états informationnels de ses locuteurs" (van Benthem 1991). Enfin, les représentations mentales des états internes sont considérées comme des expressions dans un langage formel interprétable dans un modèle du monde extérieur.

Ce paradigme, dont on ne donne ici qu'une épure grossière, est issu des réflexions de Turing en 1950 et de Putnam en 1960. Il a nourri un intense débat sur la nature de l'esprit et il s'est enrichi de travaux plus "techniques" sur les propriétés formelles des représentations mentales, sur la spécification de l'architecture fonctionnelle du système cognitif (hiérarchie de mémoires, parallélisme...) et sur la complexité des algorithmes calculant les fonctions cognitives. Ces précisions théoriques ont permis de définir et de tester expérimentalement des modèles d'apprentissage, de raisonnement, de mémorisation, de résolution de problèmes, de compréhension des énoncés.

L'approche neuronale

A cette conception "mentale" de la cognition, réunissant informatique, logique, linguistique et psychologie, se juxtapose une conception plus "biologique" qui s'attache à la description des systèmes neurophysiologiques associés à la réalisation de telle ou telle fonction cognitive : la reconnaissance des formes (visages, parole...), l'apprentissage de la coordination d'actions, etc. L'approche neuronale, dont les prémisses conceptuelles sont contenues dans les réseaux de neurones formels de McCulloch et Pitts (1947) et les automates cellulaires de von Neumann (1958), se développe aujourd'hui dans un cadre mathématique qui emprunte généralement à la dynamique des systèmes non-linéaires ou aux processus de Markov. La fécondité de cette approche alternative est manifeste en intelligence artificielle subsymbolique, en particulier pour la simulation de l'apprentissage ou de la reconnaissance. Plus important peut-être, elle stimule également les recherches sur les nouvelles architectures de machines et sur le calcul hautement parallèle. Deux questions se posent cependant quant à sa pertinence cognitive : l'adéquation des modèles de neurones et de systèmes neuronaux par rapport à leurs référents biologiques ; et dans un ordre d'idées différent, la difficulté formelle et épistémologique d'établir un lien rigoureux entre les structures mathématiques et architecturales du calcul neuronal et des fonctions cognitives essentielles comme l'interprétation des énoncés.

Peut-être n'est-il pas inutile, au moment où le développement des sciences cognitives leur donne un impact intellectuel et social considérable, de rappeler que l'informatique, comme science du calcul effectif, n'a rien fait de moins que permettre leur fondation. Il reste à argumenter qu'à bien des égards l'avenir de l'informatique passe par son dialogue avec les sciences cognitives.

Mario Borillo, directeur de recherche au CNRS, Institut de recherche en informatique de Toulouse (URA 1399 CNRS), Université Paul-Sabatier, 118, route de Narbonne, 31062 Toulouse Cedex.

L'ERGONOMIE COGNITIVE

Les logiciels modernes veulent adapter la machine à l'homme plutôt que l'homme à la machine. Leur conception s'appuie sur la psychologie cognitive.

COGNITIVE ERGONOMICS - Software engineers often produce cognitive ergonomics unwittingly. But they must be wary of simplistic psychology. By making use of the knowledge of cognitive psychology, they can avoid certain pitfalls although it is not always enough to solve all the difficulties.

André Bisseret

L'ergonomie est une technologie permettant d'adapter un "artefact" aux personnes qui doivent l'utiliser (outils, machines, locaux, meubles, véhicules...), et ceci selon des critères de confort, de facilité ou de rapidité d'usage et de minimisation des risques.

La démarche du tailleur qui prend les mesures de son client pour ensuite tailler un habit ajusté et dans lequel il se sente bien est typique de la démarche de l'ergonomie : partir de connaissances sur la personne pour spécifier l'artefact.

L'informatique aussi doit tenir compte de connaissances physiologiques pour définir les caractéristiques physiques des matériels : connaître les mensurations de la main pour définir les dimensions d'une "souris" de désignation ou celles d'un clavier ; connaître la physiologie (et pathologie) de la vue pour concevoir les écrans. Mais le logiciel est primordial pour l'utilisateur et il s'agira alors "d'ergonomie cognitive" : connaître le fonctionnement cognitif de l'homme pour y adapter le fonctionnement du logiciel. On parle aussi "d'ergonomie du logiciel".

Tel concepteur d'un logiciel, sur une maquette, fait pratiquer par un échantillon de personnes les différentes commandes prévues, mais sans les avoir nommées a priori. Il demande à chacune de choisir un libellé pour chaque commande. Sur la base des données ainsi recueillies, il choisit les mots ou expressions donnés le plus fréquemment par les personnes. Il cherche ainsi à adapter le vocabulaire de commande de son logiciel à la représentation lexicale que de futurs utilisateurs se font spontanément des actions concernées.

Un autre exemple : on montre en psychologie qu'un sujet, s'il a résolu un problème d'une certaine façon, lors d'un nouveau problème qui lui paraît du même type, va avoir tendance à essayer d'abord la méthode qui lui a déjà réussi. Sur cette base, tel concepteur de logiciel adopte une méthode unique pour la destruction de tout objet : qu'il s'agisse de supprimer une lettre, un mot, une phrase, un dessin etc. La procédure sera toujours la même : désigner l'objet, et commander "effacer".

Reconnaissance de geste au moyen du gant numérique, ici le nombre 3 (cliché LIMSI).

Optimiser l'interaction utilisateur-logiciel

De façon plus générale, on peut schématiser la problématique de l'ergonomie du logiciel de la façon suivante. L'objectif est d'optimiser l'interaction entre l'utilisateur et le logiciel, en intervenant sur la conception de ce que l'on appelle l'interface homme-ordinateur. Or il est une autre interaction primordiale : il s'agit de l'interaction entre la personne et la situation-problème dans laquelle elle se trouve.

Dans une situation donnée, la personne a deux principaux types d'activité cognitive :

- une activité de représentation : la personne, en fonction de ses objectifs, sélectionne et code certaines caractéristiques de la situation. Cette représentation est partielle, voire partiale, pouvant même être une déformation de la situation "objective";
- une activité de traitement de cette représentation. Il s'agit des ensembles organisés d'opérations, effectués pour atteindre l'objectif.

Il est fructueux de considérer la machine (l'interface) comme une représentation, choisie par le concepteur pour l'utilisateur. Elle résulte elle aussi d'une sélection et d'un codage, et ceci tant du côté présentation d'informations que du côté des commandes. Seules certaines informations sont fournies et seules certaines actions sont permises, et ceci sous certaines formes parmi les possibles. Optimiser l'interaction homme-machine, ▶

c'est assurer une bonne compatibilité entre les deux représentations de la situation : celle choisie par le concepteur et celle forgée par l'utilisateur, et ceci avant tout par une adaptation de celle-là à celle-ci.

La psychologie cognitive qui étudie l'interaction entre l'homme et son environnement, est candidate à servir de base à la conception d'interactions homme-machine bien adaptées.

Mais la pratique précède souvent la science, et l'ergonomie n'échappe pas à cette règle. Beaucoup de logiciels modernes cherchent à être des outils d'aide à l'activité cognitive de leurs utilisateurs. Leurs concepteurs font donc de l'ergonomie (éventuellement sans le savoir) : ils utilisent des connaissances sur le fonctionnement cognitif humain pour définir l'interface entre l'utilisateur et la machine. Cependant ces connaissances mises en œuvre peuvent être des connaissances fausses. De même qu'il existe une physique naïve, il existe une psychologie naïve : sur le fonctionnement intellectuel de son semblable, tout un chacun développe des croyances, souvent partagées par d'autres, et qu'il tend à admettre comme évidentes. Cette psychologie naïve se retrouve aux niveaux culturels les plus élevés.

Minimiser le recours à l'intuition

Longtemps les concepteurs de logiciel ont traité les problèmes d'ergonomie de cette façon intuitive. Ceci n'a pas donné que des échecs, loin de là, en vertu du fait qu'à créer des artefacts de façon empirique, on réussit malgré tout souvent. Ce n'est que plus tard que l'on comprend le pourquoi (des échecs comme des réussites).

A l'heure actuelle, des concepteurs d'interface de plus en plus nombreux minimisent le recours à l'intuition et cherchent à utiliser des connaissances obtenues par la psychologie cognitive.

Il restera que la conception d'une interface (comme de tout autre artefact) ne se déduit pas directement de résultats scientifiques. L'ergonomie, comme toute ingénierie, comporte une part d'art. Chercher cependant à étayer les choix, autant que possible (et entre autres critères), par des connaissances de la psychologie ne peut qu'augmenter les chances de concevoir des logiciels mieux adaptés à leurs utilisateurs.

Quelques développements actuels peuvent être soulignés, pour lesquels l'ergonomie est cruciale et auxquels peuvent contribuer les sciences cognitives.

On cherche à dépasser les interfaces qui n'autorisent que la commande par clavier et souris. Les progrès de la reconnaissance de la parole devraient permettre d'adjoindre la commande vocale, du moins dans des applications simples à langage limité. La commande par gestes est déjà réalisée en laboratoire et à condition de porter un gant spécial, muni de capteurs des mouvements. On commence à envisager des commandes par gestes, captés par caméra puis interprétés par l'ordinateur. De nombreuses équipes travaillent ainsi sur ce qui est appelé le "multimodal".

L'utilisation par l'ordinateur du langage naturel général et complet n'est pas pour demain. Mais on montre que pour peu qu'elles se spécialisent dans une tâche précise, les personnes construisent et utilisent entre elles des langages naturels mais restreints, et relativement contraints (sous-langages, langages opératifs) qui eux pourront être utilisés bientôt pour les dialogues homme-machine.

Enfin, un domaine très actif est celui de l'aide à la conception : conception assistée par ordinateur (mécanique, architecture...), à laquelle il faut ajouter la conception de logiciels, de textes, d'images, d'animations etc. Le défaut principal des systèmes d'aide actuels est d'avoir été spécifiés à partir d'une analyse du produit fini (dessin, programme, texte...) plutôt que d'une analyse du processus de conception de ce produit. Or, une intense activité se déroule actuellement, tant en recherche sur ces activités humaines de conception qu'en développement de nouveaux concepts de systèmes d'aide, cette fois grâce à une stratégie radicalement "centrée-utilisateur".

Dans une telle stratégie de conception des interfaces, la psychologie et plus généralement les sciences cognitives s'ajoutent aux mathématiques comme disciplines de base de la technologie informatique.

André Bisseret, directeur de recherche à l'INRIA, IRIMAG-Laboratoire de génie informatique, BP 53X, 38041 Grenoble Cedex.

La robotique avancée

Les robots de troisième génération, sur lesquels portent les recherches actuelles, seront "intelligents", c'est-à-dire qu'ils seront dotés de capacités de perception, de modélisation, de raisonnement, de communication et d'action. Ils mettront en œuvre les lois de la physique, des mathématiques, de l'informatique, de l'intelligence artificielle, du traitement du signal, de l'automatisme et de l'ergonomie.

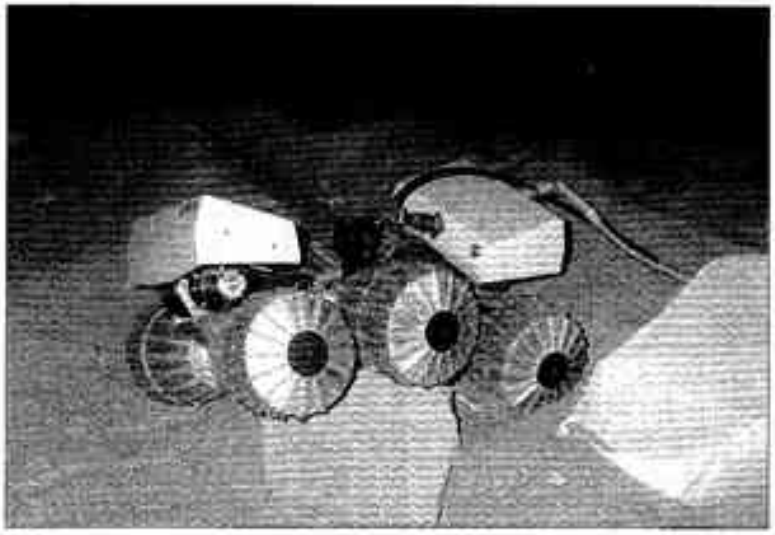
Pour les mettre au point, il est nécessaire de passer par une conceptualisation, non seulement des propriétés du robot, mais aussi des caractéristiques de l'environnement. Le concepteur est par ailleurs conduit à fractionner et décomposer les problèmes et à les résoudre en intégrant des blocs fonctionnels élémentaires. On retrouve donc en robotique avancée la complexité des systèmes multi-agents et des bases de données.

Les caractéristiques du monde physique dans lequel le robot doit travailler sont déterminantes. Il faut ainsi distinguer les environnements structurés (routes, ateliers, centrales nucléaires, espaces aériens et aérodromes, espace interplanétaire), des milieux non structurés (fonds marins, surface des planètes, espaces agricoles, champs de bataille...) qui sont plus variés et inattendus.

De nouvelles recherches (dont la France est plutôt absente) s'intéressent aux robots minuscules dont l'environnement a une échelle microscopique. Leur champ d'application pourrait inclure l'observation des organes vivants et des opérations chirurgicales in situ.

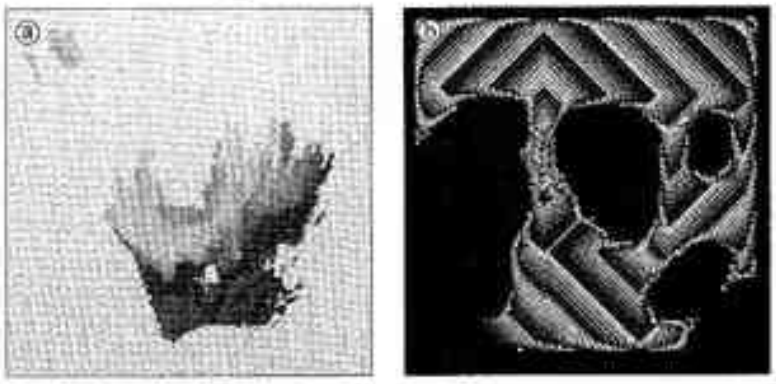
Pour percevoir l'environnement, les robots de troisième génération utilisent des capteurs qui fournissent des signaux donnant des informations sur le milieu où ils se trouvent. On choisit pour la vision, selon les cas, des caméras CCD (*charge coupled device*), des télémètres lasers à balayage, des radars, des sonars... Le traitement en temps réel des images ainsi obtenues permet d'obtenir l'équivalent d'une véritable vision artificielle.

La vision 3D statique permet de déterminer la géométrie tridimensionnelle de la surface des solides. Elle peut utiliser diverses configurations de caméra. Des multiplets d'images à deux dimensions permettent de reconstruire, par corrélation, les données tridimensionnelles (vision monoculaire multifocale, vision stéréoscopique binoculaire ou même trinoculaire). Les capteurs actifs, tels que le télémètre laser à balayage ou le sonar en milieu liquide, donnent directement une information tridimensionnelle.



Marsokhod : véhicule de l'Institut Transmach de St Petersbourg, devant participer à la mission d'exploration MARS 96.

Dans les environnements structurés (en général artificiels), les images peuvent être réduites à des segments "significatifs" qui correspondent souvent aux arêtes de polyèdres et au contour apparent des objets composant la scène, ce qui réduit beaucoup le volume des calculs.
Dans les milieux non structurés où de telles simplifications ne sont plus possibles, les recherches sont moins avancées. La solution connue (employée par exemple pour l'observation stéréoscopique de la Terre par des satellites d'observation) passe par le calcul complet de la fonction d'intercorrélation des images et est trop lourde en calcul pour une application "temps réel".
La vision "dynamique" combine perception de scène et perception du mouvement. Une solution évidente utilise la vision statique avec des cadences élevées de répétition et des temps de traitement très courts. Il existe d'autres solutions : l'existence du mouvement permet d'utiliser les images successives d'une caméra unique comme des données quasi stéréoscopiques et le "flot" visuel contient des informations sur le mouvement.
La fusion sensorielle a pour but de fournir une représentation cohérente et unique à partir des données provenant de divers capteurs. On peut, par exemple, fusionner une information monoscopique et des mesures éparses de télémétrie, une séquence d'images successives, les informations de distance et de luminance d'un télémètre imageur, les données de la vision dynamique et des informations de navigation inertielle.
La fusion a un caractère probabiliste, aussi faut-il prendre en compte les incertitudes et déterminer celles-ci ainsi que le taux de confiance affecté au résultat.
Le robot est par ailleurs constitué d'un ensemble d'actionneurs dont l'activation coordonnée et contrôlée va permettre l'exécution de mouvements. Cette fonction est d'une complexité très variable : un robot mobile à roues est notablement plus simple à modéliser qu'un robot quadrupède.
L'interaction avec l'environnement qui résulte de l'activation dynamique du robot est difficile à modéliser finement, en particulier pour des tâches de locomotion ou de saisie d'objet. Une solution prometteuse consiste à doter le robot de capacités d'action réflexes assurées par des capteurs spécifiques caractérisant localement l'interaction du robot et de l'environnement. Le contrôle réflexe ou réactif agit alors sur la base d'un modèle local, simple, de l'interaction.
A tout moment, un robot intelligent doit être capable d'adapter les paramètres de réglage et de sélectionner la combinaison des ressources dont il dispose, en fonction de ce qu'il a perçu, pour accomplir la tâche qui lui a été fixée. Il doit donc élaborer un plan et même envisager l'échec de ce plan pour en adapter un autre, en fonction de l'expérience acquise. Il est inutile de dire qu'il s'agit là d'un problème très difficile. Le contrôle d'exécution opère une rétroaction sur le programme du robot, à la manière d'une boucle d'asservissement.
Le robot est souvent en contact avec un opérateur humain qui peut se contenter de le surveiller, mais qui peut aussi le télécommander. Plus le robot est autonome, plus le programme téléchargé par l'opérateur humain se limitera à une description de la mission, sans entrer dans les détails d'exécution.
La variété des algorithmes envisageables, la diversité des connaissances pertinentes et la multiplicité des architectures possibles donnent à penser que le robot intelligent a des perspectives d'évolution inépuisables. Si, sous sa forme actuelle, il reste marqué par l'apport de l'intelligence artificielle, sa conception pourrait évoluer profondément avec l'emploi des réseaux neuronaux et les diverses approches connexionnistes qui devraient renouveler les problématiques de réactivité et d'apprentissage.



Exemples d'élaboration de modèles topologiques et de cartes pour un robot mobile autonome : a) carte d'élévation mesurée; b) carte de navigation. (© Matra Marconi Space).

Jean-Louis Lacombe, directeur scientifique délégué de Matra Marconi Space.

LA PERCEPTION VISUELLE DU ROBOT

La perception des objets, que l'on croyait facile dans les années 60, s'est révélée très difficile. L'approche moderne suppose une mathématisation poussée qui ne permet aujourd'hui que la perception de phénomènes simples.

Olivier Faugeras
Roger Mohr

THE ROBOT'S VISUAL PERCEPTION – Contrary to hopes held in the 60's, it has turned out to be very difficult to build a machine capable of perceiving its physical environment in the way human beings do. We have yet to find a method for gleaning descriptions of visual objects from images which an artificial intelligence program could use. Today, the best results are obtained with methods based on mathematical modelling of such objects. But they can only be applied to relatively simple, well-defined objects.

L'un des buts de la vision artificielle est de construire des représentations de l'environnement à partir des informations fournies par des capteurs. Ces représentations peuvent être implicites ou explicites. Elles doivent permettre au système informatique, parfois assisté par un opérateur humain, d'exécuter les actions convenables. Les calculs qui conduisent à la formation de la représentation doivent être très rapides et la réponse quasi instantanée s'il s'agit de piloter un véhicule ou de conduire une opération d'assemblage à l'aide d'un bras articulé. Le système doit intervenir alors de la même manière que dans la vision animale ou humaine, pour permettre une action immédiate de locomotion ou de manipulation.

Mais il existe aussi des cas où le facteur temps n'est pas aussi impératif. L'utilisation des images prises par satellite pour établir une carte topographique peut exiger un traitement de longue durée sans inconvénient majeur.

Mathématiques ou intelligence artificielle

Deux voies de recherches, utilisant des moyens différents, se sont dégagées dès les premières études de vision par ordinateur. La première, se fondant sur les mathématiques et la physique, s'attachait à comprendre la formation de l'image. La seconde, issue de l'intelligence artificielle, souhaitait comprendre la scène sous forme symbolique. Il est amusant de penser que dans les années 60 le problème n'était pas considéré comme difficile. Marvin Minsky avait même proposé à quelques-uns de ses étudiants d'en faire le thème de leur projet de fin d'année scolaire.

La faible puissance de calcul disponible à l'époque paraissait le seul obstacle aux réalisations. Cette illusion est aujourd'hui dissipée : les difficultés demeurent malgré les moyens de calculs modernes.

Les travaux importants menés depuis cette date ont conduit à de nombreux résultats intéressants. Les techniques de propagation de contrainte, appliquées aujourd'hui dans les systèmes à maintenance de connaissance en intelligence artificielle, sont issues de travaux sur la vision. Mais les chercheurs ne sont pas parvenus à trouver des modèles corrects de représentation symbolique.

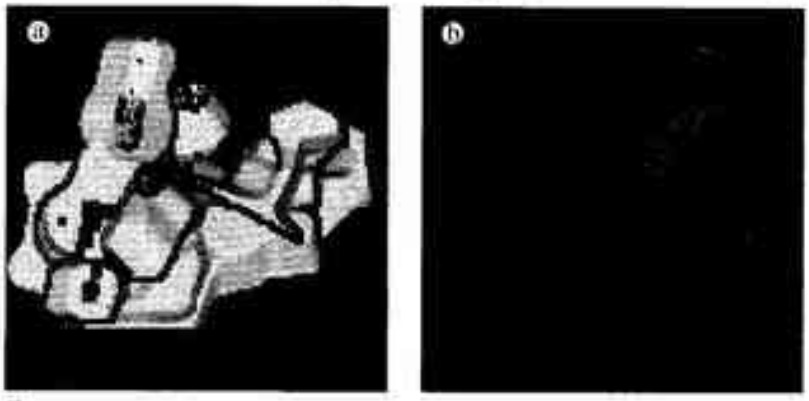
Durant les années 70, avec le développement des premiers outils de représentation des connaissances, des modèles relationnels simples ont été proposés. Il s'agissait d'exprimer des contraintes du type "une table à quatre pieds verticaux qui la supportent". On espérait, en affirmant de telles structures, définir des termes génériques "d'objets" permettant d'analyser des scènes complexes comme un intérieur d'habitation. Il a fallu se rendre compte que ces modèles simples ne se généralisaient qu'au prix d'efforts disproportionnés et que des outils de perception de bon niveau éprouvaient les plus grandes difficultés à reconnaître "un objet fin, à peu près rectiligne, et vertical".

Ce n'est pas qu'un problème de puissance de calcul

Ni le parallélisme massif des ordinateurs les plus récents, ni un contrôle intelligent n'avaient su répondre aux questions qui se posaient. A la suite de l'échec relatif de la voie de l'intelligence artificielle, l'orientation mathématique a pris un essor important ces dix dernières années.

Selon cette méthode, les problèmes de vision sont posés comme des problèmes "possibles" au sens de la théorie mathématique et informatique de la complexité. Les contraintes sur la tâche à accomplir, les caractéristiques des capteurs sont utilisées pour réduire la complexité de la tâche de perception en limitant les domaines de variations des grandeurs en jeu, ce qui facilite le traitement combinatoire.

La modélisation de la réflectance avec application à l'identification des surbrillances et à la modélisation géométrique des objets conduisant à leur identification est un exemple de cette problématique. Ce domaine passionnant est en évolution très rapide. Il est intéressant de noter qu'il correspond à une approche très différente de celle, plus classique, qui tend à copier les systèmes biologiques, dont il est difficile d'imagi- ▶



a) Image tridimensionnelle d'un vrac d'objets : l'intensité code l'orientation de la normale à la surface des objets ; en clair, la normale est dirigée vers l'observateur, en sombre, elle s'éloigne de lui.
b) Les résultats de l'analyse de l'image a par un programme de reconnaissance et de positionnement : en bas à gauche (bleu), le modèle tridimensionnel utilisé ; en haut à gauche, le découpage de la scène à en facettes planes ; en bas à droite, le modèle dans l'orientation dans laquelle il a été reconnu dans la scène a, avec en vert les facettes effectivement identifiées ; en haut à droite, la superposition du modèle (rouge) et des facettes de la scène reconnue (vert).

ner qu'ils utilisent des modèles aussi sophistiqués de réflectance pour percevoir l'environnement.

Une telle formalisation, tournée vers une modélisation quantitative, contraste avec l'approche symbolique qui avait été tentée précédemment. Elle implique une mathématisation très poussée de la discipline. La géométrie lie les observations tridimensionnelles et leurs projections perspectives. La physique de la lumière sert à modéliser les capteurs et le processus de formation des images. Le traitement du signal permet de distinguer les textures et les contours. L'alliance de la géométrie et de la cinématique conduit à la perception du mouvement.

Choisir entre des informations contradictoires

Les différents outils ainsi mis en place sont complémentaires. Mais la redondance des informations qu'ils fournissent cache un piège. Les informations ne sont pas parfaites et sont affectées d'un "bruit". Il arrive donc que des informations qui devaient être complémentaires se révèlent contradictoires. Par exemple, la mise en œuvre simultanée d'un détecteur de contour et d'un détecteur de régions est une méthode très efficace si le contour repéré sépare bien les régions. Mais que faut-il faire quand – et cela arrive souvent – le détecteur de contour délimite deux régions, alors que le détecteur de régions n'en trouve qu'une seule ?

Trouver une solution cohérente relève alors de l'intelligence artificielle. Mais il est aujourd'hui prématuré de vouloir prédire quelle est la technique qui sera la plus efficace. Les méthodes déductives l'emporteraient-elles sur les méthodes plus globales comme la relaxation ou l'utilisation des processus de décisions liés aux réseaux de neurones ? Personne ne le sait aujourd'hui.

Actuellement, certains problèmes simples comme la perception de la trajectoire d'une balle de ping-pong pour permettre à un robot de jouer à ce jeu sont à notre portée, parce qu'ils se mathématisent assez facilement. Mais identifier l'expression d'un visage, découvrir un objet dont la forme est approximativement connue sont hors d'atteinte des formalisations actuelles.

Les perfectionnements attendus pour la prochaine décennie concernent surtout la perception alliée au mouvement : reconstruction tridimensionnelle, perception des mouvements des objets, mouvements non rigides. En revanche, tout reste à inventer dans l'identification et la localisation dès que les dimensions et la forme de l'objet ne sont pas parfaitement définies. Les techniques neuronales, grâce à leurs capacités d'apprentissage, apporteront peut-être une solution. Mais les capacités d'abstraction actuelles de tels systèmes sont encore insuffisantes pour qu'elles puissent opérer sur une image, même prétraitée.

Olivier Faugeras, directeur de recherche à l'INRIA, maître de conférences à l'Ecole polytechnique. INRIA, 2004, route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex.

Roger Mohr, professeur à l'Institut national polytechnique de Grenoble, Laboratoire d'informatique fondamentale et d'intelligence artificielle (URA 394 CNRS), 46, rue Félix Viallet, 38031 Grenoble Cedex.

LE ROBOT QUI DÉCIDE SEUL

Un robot autonome devrait être capable de planifier lui-même les tâches qui lui sont confiées. Le système de planification qui décide des actions à entreprendre et de leur organisation est particulièrement complexe.

Malik Ghallab

AUTONOMOUS ROBOTS - Telling a robot the goal it must reach and letting it decide on how to achieve it is very ambitious. The state operator approach is too restrictive to be practicable. But the necessary extensions involve prohibitively complex computations. Compromises must be found.

Le réflexe est immédiat, mais l'action délibérée nécessite un raisonnement préalable et un plan qui la situe logiquement pour l'achèvement d'un but. Faire preuve d'autonomie requiert une capacité à agir dans des situations nouvelles, pour lesquelles des réflexes seuls sont insuffisants. Un robot autonome doit donc être capable d'élaborer son plan d'action. Il connaît le but à atteindre et la situation où il se trouve. Il dispose d'une description de tout ce qu'il peut faire, sous la forme de modèles abstraits d'actions : quand une action est réalisable et quels sont ses effets. Il doit trouver un plan, soit un ensemble d'actions organisées dans le temps, qui lui permettra, si tout se déroule comme prévu, de passer de la situation présente au but fixé.

Le terme de planification peut avoir en informatique trois interprétations très différentes, de complexité croissante :

- la gestion de projets : on connaît les actions à réaliser et leurs contraintes d'organisation, mais on ne dispose que d'informations imprécises sur leur durée. Il s'agit de définir et de gérer un planning : dates butoirs et tâches critiques dont le retard serait pénalisant ;
- l'ordonnancement de tâches : on connaît les actions à réaliser mais pas leur organisation, ni les ressources à leur attribuer. Il s'agit de trouver une organisation et une utilisation efficace des ressources ;
- la synthèse de plans : on ne connaît ni les actions, ni leur organisation. On dispose d'une description de tout ce qu'il est possible de faire, de l'état courant et du but à atteindre.

Un système capable de faire la synthèse de plans

Pour la réalisation d'un robot autonome, c'est bien entendu d'un système de synthèse de plans dont on a besoin. Un tel système peut être général, ou bien spécifique à un environnement ou à une tâche particulière, telle que par exemple la planification de mouvements pour la manipulation ou la navigation. Des systèmes de planification spécifiques sont bien entendu nécessaires à un robot, mais pas toujours suffisants du fait de la variabilité des tâches et des environnements, et du degré de flexibilité et d'autonomie souhaité pour le robot. Le cas général pose des problèmes multiples. Le système doit être capable :

- de représenter formellement une action, en tant que changement dans le monde ;
- de raisonner efficacement sur cette représentation ;
- de mettre en correspondance les résultats prédits par le raisonnement avec les effets observés des actions effectuées, car on ne peut s'attendre à une conformité stricte aux prévisions, qui sont nécessairement incomplètes et approximatives ;
- d'intégrer l'activité du robot à l'évolu-

tion propre de son environnement, prévue ou perceptible :
- de prévoir dans le plan des actions de perception, qui positionneront ou adapteront les senseurs à la tâche ;
- de gérer des actions de communication, par exemple pour coordonner l'activité avec celles d'autres opérateurs ou robots, dans la cas d'une tâche distribuée.

Représentation de l'action par des opérateurs d'état

Des travaux relativement récents ont permis une caractérisation complète et satisfaisante de la représentation de l'action dite des opérateurs d'état. Cette représentation remonte au début des années 70 au système STRIPS. On y décrit une action par un ensemble de pré-conditions et un ensemble de post-conditions, chacune étant une proposition logique. La situation initiale et les buts à atteindre sont des ensembles de propositions. La situation résultant d'une action s'obtient en ajoutant les post-conditions à la situation de départ, et en enlevant de celle-ci les propositions en contradiction avec les post-conditions. On cherche un ordre partiel d'actions permettant de passer de la situation initiale à une situation incluant l'ensemble des propositions du but. Les actions sont des transitions instantanées entre états ; elles ne peuvent se recouvrir ni s'effectuer en parallèle. L'ordre partiel recherché ne prend pas en compte le parallélisme, il est simplement équivalent à une famille de séquences d'actions.

Ce problème, bien que complexe en théorie, admet une procédure complète et relativement efficace en pratique, car elle repose sur un algorithme de vérification de plan polynomial. La vérification consiste à déterminer quelles propositions seront satisfaites à chaque étape du plan ; c'est donc une opération cruciale pour décider de l'applicabilité d'une action.

Cette représentation a donné lieu au développement de plusieurs prototypes. Elle est cependant trop restrictive pour une application conséquente. De nombreuses extensions sont nécessaires, parmi lesquelles :
- la prise en compte des "ramifications de l'action" : on ne peut décrire explicitement tous les effets d'une action, certains effets seront implicites ou dépendront du contexte, ils devront pouvoir être déduits à partir d'axiomes généraux ou spécifiques au domaine (par exemple le déplacement d'une boîte entraîne le déplacement de tout ce qu'elle contient) ;
- la prise en compte des "qualifications de l'action", c'est-à-dire les très nombreuses conditions qui permettent ou empêchent son application. Seul un petit nombre de ces conditions est exprimable dans les pré-conditions, les autres sont supposées satisfaites par défaut (par exemple, la boîte à transporter est supposée ordinaire, non fragile, ne comportant pas un produit liquide ou volatil ou explosif exigeant une action spécifique) ;
- la prise en compte d'actions conditionnelles dont les effets varient selon les contextes, ou d'actions combinées dont les effets non additifs sont très différents des effets de chacune des actions individuellement (par exemple, le maintien par une "main gauche" pour assurer équilibre, position ou effort pendant la manipulation par une main droite) ;
- la prise en compte du temps dans la durée des actions, dans leurs positions respectives ou relativement à des événements extérieurs ou à des buts associés à des délais.

Malheureusement, la plupart de ces extensions utiles augmentent considérablement la complexité de la tâche de planification. Par exemple, la prise en compte d'effets dépendant du contexte ou des ramifications de l'action rend la vérification de plan de complexité non polynomiale, or ce problème doit être résolu à chaque étape élémentaire de planification. Ceci ne décourage pas les recherches, car des compromis satisfaisants pour la résolution de ces problèmes ne sont pas exclus.

Extensions et compromis

Une approche, appelée "calcul situationnel", a été proposée initialement pour déduire tous les effets d'une action. Mais elle a conduit au "problème de rémanence" : de nombreux axiomes sont nécessaires pour spécifier non seulement ce qui change, mais également les faits invariants dans un changement. Ce problème est évité dans l'approche restrictive des opérateurs d'état : tout est invariant sauf ce qui est explicitement mentionné dans le modèle de l'action. Plusieurs alternatives considèrent que le résultat de l'action A est l'état "le plus proche" de l'état courant, et dans lequel toutes les conséquences explicites de A sont satisfaites. Se pose alors le problème de définir et de calculer cet état le plus proche. Les recherches sur ce sujet sont très actives et rejoignent des préoccupations de raisonnement non monotone et de révision de croyances.

La prise en compte explicite du temps est également un axe de recherches actives en planification. Certains systèmes ajoutent à l'approche classique une gestion de contraintes temporelles (durées, fenêtres d'occurrences par exemple). D'autres approches reposent sur des logiques temporelles dites réifiées, où chaque proposition est qualifiée par un intervalle temporel. Ces représentations sont plus riches et correspondent mieux aux besoins d'un robot. Elles ne résolvent pas pour autant les autres extensions souhaitables : ramification, qualification, ou actions conditionnelles.

Mentionnons pour conclure les travaux qui proposent de fournir a priori au robot une famille de plans, couvrant toutes les tâches qu'il aura à réaliser, et suffisamment flexibles pour être utiles malgré une variabilité limitée de l'environnement. Ces plans, dits universels, sont très attrayants pour leur réactivité. Ils sont nécessaires à un robot et définissent ses réflexes, mais ils ne lui confèrent qu'une faible autonomie. Leur utilité est plus grande s'ils sont associés à un système de planification général et à un mécanisme d'apprentissage.

La famille des robots HILARE : trois générations vers davantage d'autonomie (LAAS CNRS)

Malik Ghallab, directeur de recherche au CNRS, Laboratoire d'automatique et d'analyse des systèmes (UPR 8001 CNRS), 7, avenue du Colonel Roche, 31077 Toulouse Cedex.

GARER UN ROBOT MOBILE

Des résultats récents montrent qu'il est possible de concevoir un programme informatique de commande d'un robot mobile semblable à une voiture, qui lui permette de se garer automatiquement, même dans des espaces très contraints.

Jean-Paul Laumond
Jean-Daniel Boissonnat

PARKING A CAR-LIKE MOBILE ROBOT - The configuration of a car-like mobile robot is defined by three parameters: two parameters for its position and one parameter for its direction. Nevertheless we get usually only two controls to steer the vehicle: the accelerator and the driving wheel. We know now how to plan feasible trajectories that not only respect the kinematic constraints of the vehicle, but also avoid the obstacles.

Chacun de nous sait garer sa voiture en faisant un "créneau". Cette manœuvre délicate a demandé un apprentissage plus ou moins long. Dans un proche avenir, une voiture automatisée sera-t-elle capable d'effectuer la même opération ? La réponse est, depuis peu, affirmative. Considérons la situation illustrée par la figure 1a.

Des mouvements sans collision

La voiture doit entrer dans le garage et se placer sur un pont élévateur situé dans le coin en haut à droite. Comment planifier une trajectoire qui amène le véhicule au but, tout en évitant les obstacles ; ou, le cas échéant, comment prouver qu'il n'en existe pas ?

Localiser le véhicule dans son environnement nécessite la donnée de trois paramètres. Par exemple deux paramètres pour la position du point milieu de l'essieu arrière et l'angle que fait le véhicule avec une direction de référence. Ces trois paramètres sont représentés par un point dans un espace de dimension trois : l'espace des configurations. L'espace des configurations sans collision (Fig. 1b) est ainsi un volume de cet espace, correspondant aux positions où le véhicule ne pénètre pas les obstacles.

Dans l'exemple pris ici, le volume a été calculé automatiquement à l'aide d'un logiciel basé sur des études de géométrie algorithmique et développé à l'INRIA. Il est possible de calculer un chemin entièrement contenu dans ce volume : par construction, ce chemin, s'il était suivi par le véhicule, éviterait les obstacles de l'environnement (Fig. 2a). Il a été démontré qu'il existe des solutions générales et exactes, issues de la géométrie algébrique réelle pour tous les problèmes de ce type, même dans les cas complexes : véhicule tractant une remorque, robot manipulateur constitué de nombreux corps articulés...

Des mouvements faisables...

Dans notre exemple, le volume de l'espace des configurations sans collision est d'un seul bloc, ce qui signifie qu'il sera toujours possible de passer d'une position à une autre sans collision. Mais n'importe quel chemin n'est pas possible dans la réalité, il faut tenir compte des contraintes de roulement du véhicule.

L'espace des configurations du véhicule est de dimension trois. Si l'on veut explorer cet espace, il peut sembler naturel de conclure à la nécessité d'avoir la

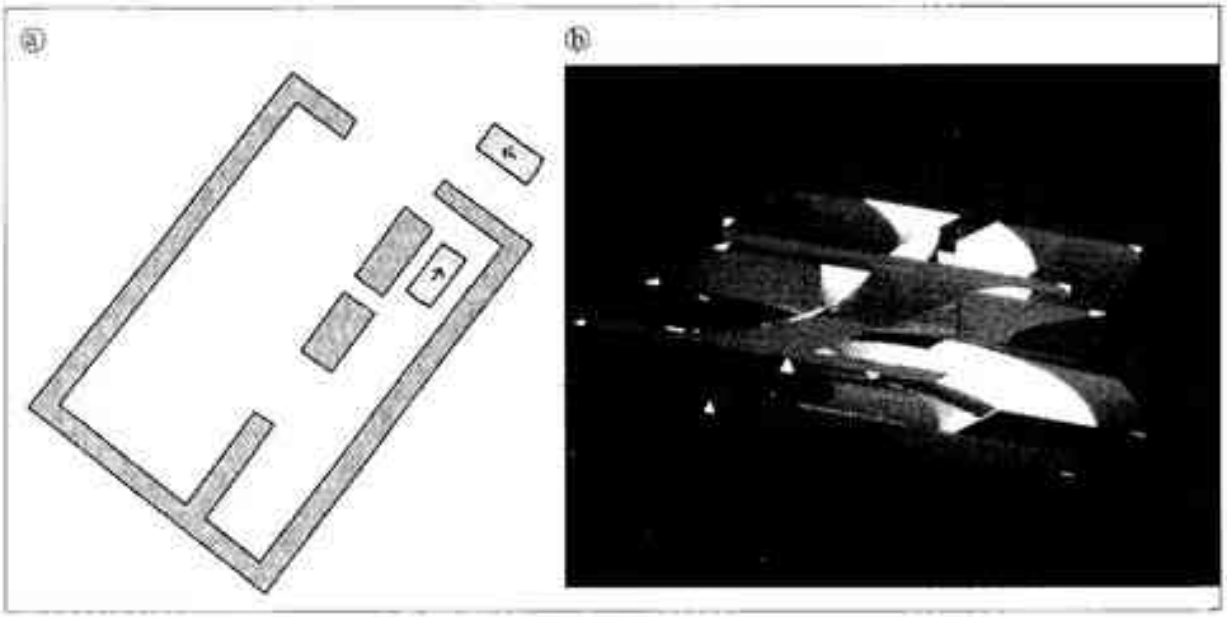


Fig. 1 - Le robot dans son environnement réel (a) et l'espace des configurations sans collision associé (b). (Résultat produit par un logiciel développé par F. Avnaim).

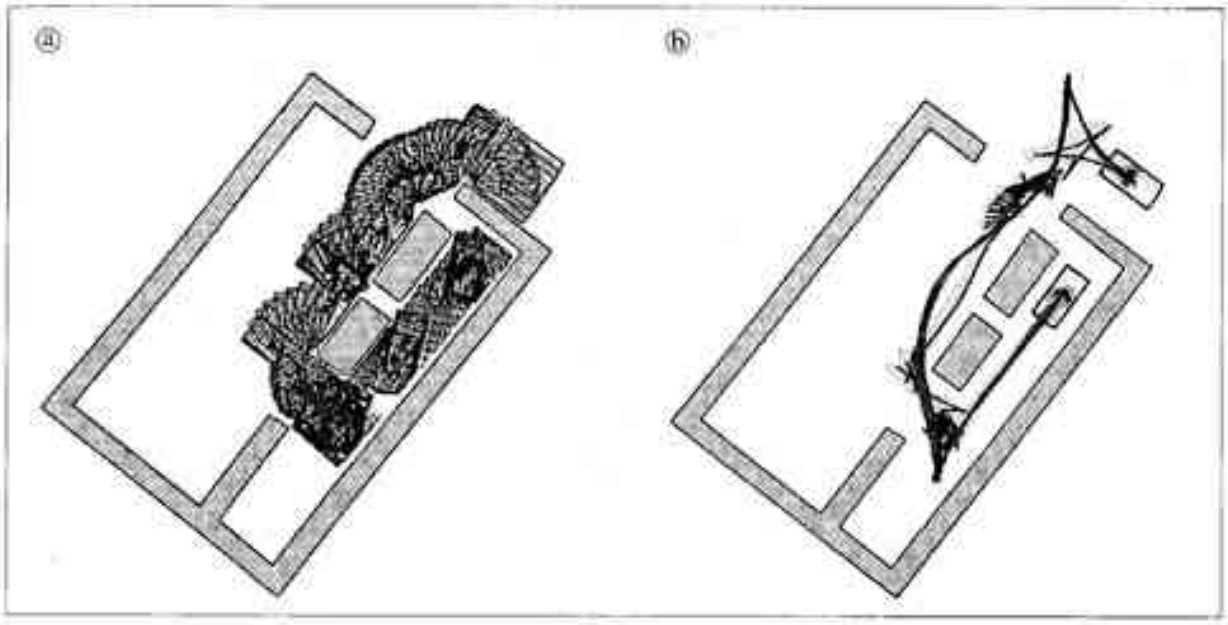


Fig. 2 - a) en vert : un chemin initial qui évite les obstacles mais n'est pas faisable par le robot ; b) - en bleu : un premier chemin faisable très proche du chemin initial, en rouge : la solution finale. (Résultat produit par un logiciel développé par M. Taïx et P. Jacobs.)

capacité de bouger dans trois directions (on parle de degrés de liberté). En fait, quand on conduit une voiture, on agit uniquement sur la vitesse du véhicule, par l'intermédiaire de l'accélérateur et du frein et sur sa direction, par l'intermédiaire du volant : une voiture n'a que deux degrés de liberté, bien qu'elle évolue dans un espace de configurations de dimension trois ; on dit que le système est non holonome. Par exemple, il n'est pas possible de faire rouler une voiture dans une direction parallèle à son essieu arrière. Pourtant, l'expérience montre qu'on peut aller où l'on veut en voiture, même dans la direction qui semble interdite ; c'est le cas du créneau, lorsqu'on gare sa voiture entre deux autres le long d'un trottoir. Le point essentiel est qu'il est en effet possible de bouger dans cette direction, mais pas directement ; ce n'est possible qu'en combinant des petits mouvements élémentaires de marche avant, marche arrière et rotation. On dit que le système est contrôlable.

La théorie du contrôle fournit les outils généraux permettant de modéliser ces combinaisons de manœuvres pour des systèmes plus généraux qu'une voiture, et de tester si elles permettent de "récupérer" les directions de mouvements a priori interdites. Des résultats récents montrent qu'une voiture à laquelle on accrocherait un nombre quelconque de remorques, comme les engins de convoyage de bagages dans les gares et aéroports, constitue un système contrôlable ; cela signifie qu'il est possible de parquer un tel engin entre deux autres le long d'un mur, comme on le fait avec une simple voiture et parfois avec une voiture tractant une caravane... Le nombre de manœuvres pour accomplir une telle tâche est bien plus considérable, mais il est possible de l'accomplir.

...et sans collision

A ce stade de la présentation, nous savons qu'il existe des solutions au problème initial, mais nous n'avons pas encore de méthode pour en calculer une. Pour y parvenir, une méthode a été récemment développée au LAAS. Elle constitue le premier algorithme exact et complet permettant de résoudre la planification de mouvements sans collision pour un système non holonome.

La méthode repose sur un résultat récent, établi par deux chercheurs américains (Reeds et Shepp) et permettant un calcul effectif de la trajectoire la plus courte pour une voiture entre deux configurations choisies arbitrairement, ceci en l'absence d'obstacles dans l'environnement. Considérons un premier chemin non faisable (Fig. 2a). L'algorithme consiste à construire une approximation de ce chemin par de petites portions de trajectoires de Reeds et Shepp qui sont sans collision avec les obstacles.

Pour cela, on choisit sur le chemin initial des configurations proches les unes des autres ; on calcule le plus court chemin entre deux configurations consécutives ; s'il est sans collision, on le conserve ; sinon on introduit une configuration supplémentaire. Ainsi, petit à petit, on est amené à calculer des portions de trajectoires entre des configurations de plus en plus proches. Intuitivement, on sent qu'à force de subdiviser, ces bouts de trajectoires vont se rapprocher de plus en plus du chemin initial ; ils finiront ainsi par tous être sans collision. On prouve effectivement que la procédure de subdivision s'arrête après un nombre fini d'étapes. La concaténation de toutes ces trajectoires élémentaires constitue une première solution au problème ; elle comporte un nombre important de manœuvres (chemin en bleu figure 2b).

Une dernière étape permet de raccourcir la longueur de la première trajectoire solution et de réduire le nombre de manœuvres (chemin en rouge figure 2b). L'ensemble de tous ces calculs prend quelques minutes sur une station de travail standard.

Jean-Paul Laumond, chargé de recherche au CNRS, Laboratoire d'automatique et d'analyse des systèmes (UPR 8001 CNRS), 7, avenue du Colonel Roche, 31077 Toulouse Cedex.

Jean-Daniel Boissonnat, directeur de recherche à l'INRIA, 2004, route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex.

COMMENT FAIRE SAISIR UN OBJET PAR UNE MAIN DE ROBOT

Pour saisir un objet il faut sentir de quelle manière on le tient et savoir ce qu'on va en faire.

Christian Laugier
Jocelyne Troccaz

***How can a robot be made to grasp an object** - It has become clear that grasping an object is not only "tactical" involving the actual act of seizing something and the fundamental interaction that implies between action and tactile-kinesthetic perception, but also "strategic", in connection with the end purpose of that act. These studies show that the concept of manual dexterity will be an essential feature of tomorrow's robot.*

La préhension humaine met en œuvre une interaction permanente entre le geste et la perception tactilo-kinesthésique. Le geste de préhension d'un objet peut être décomposé en quatre phases : un mouvement balistique vers l'objet accompagné d'une préconfiguration de la main, un mouvement d'approche terminal apte à prendre en compte les contraintes d'encombrement de l'espace environnant, de petits mouvements de la main et des doigts permettant d'ajuster ceux-ci sur l'objet à saisir, et enfin d'éventuels mouvements des doigts permettant de repositionner l'objet tenu lorsque cela est rendu nécessaire par la tâche à exécuter (par exemple : saisie d'un crayon posé sur une table afin de pouvoir écrire). Les deux premières phases sont guidées par des informations visuelles, la troisième phase fait surtout appel à des données de type tactilo-kinesthésique. La quatrième phase est la plus complexe et met en jeu l'ensemble des mécanismes perceptifs mentionnés plus haut.

Sur le plan conceptuel, les différentes phases combinent des problèmes de nature tactique (quelles sont les modalités des mouvements mis en jeu ?) avec des questions de nature stratégique (quelles parties de l'objet faut-il saisir et suivant quelle stratégie ?). Toute la difficulté consiste à transposer ce schéma décisionnel au niveau des mécanismes de contrôle d'un robot équipé d'un organe de préhension évolué (typiquement : une main articulée équipée de capteurs tactiles, voir figure).

Composante tactique et composante stratégique

La composante tactique met en jeu des techniques informatiques qui relèvent de l'algorithmique du mouvement et de l'automatique. Il faut raisonner sur les contraintes de déplacement du préhenseur, comme des obstacles ou des butées mécaniques et sur les caractéristiques de la boucle commande/perception. Cette composante du problème a retenu l'attention des chercheurs dans les années 80. Des techniques algorithmiques ont été développées, soit comme solutions particulières au problème de la planification de trajectoires sans collisions, soit comme outils spécialisés venant en complément de techniques d'analyse de scènes afin de pouvoir sélectionner automatiquement des caractéristiques appropriées sur les objets à saisir dans un vrac. Parallèlement, des progrès spectaculaires portant sur la mécanique et sur la commande des préhenseurs ont permis de faire émerger le concept de "dextérité manuelle" qui constitue l'une des briques de base de la robotique de demain.

La composante stratégique met en jeu des raisonnements liés au concept "d'intelligence du geste". Elle s'appuie sur des connaissances opératoires et sur une analyse géométrico-physique du problème à résoudre. Cette analyse conduit à évaluer des critères variés tels que la morphologie de l'objet, ses dimensions en regard de celles du préhenseur et la stabilité de l'ensemble. On raisonnera par exemple sur des caractéristiques du type : l'objet est plutôt gros, petit, long, mince, plat, constitué de plusieurs protubérances... On prendra également en considération des contraintes de stabilité, de précision et d'accessibilité qui sont directement liées à la tâche à réaliser (par exemple : prise en force pour la saisie d'un marteau, ou prise de précision pour une aiguille). Ceci se traduit sur le plan pratique par le choix d'une partie à saisir sur l'objet, et par la sélection d'une direction d'approche et d'une "préconfiguration" appropriée du préhenseur de manière à préparer la saisie effective. Cette manière de faire se justifie par trois constatations :

- on ne peut retenir qu'un petit nombre de configurations : moins de vingt pour une main humaine ;
- les configurations de saisie ont des caractéristiques similaires à celles des préconfigurations sélectionnées ;
- le choix d'un positionnement particulier des doigts dépend essentiellement de la nature des contacts et des forces mis en jeu (ce qui suggère un ajustement réactif conditionné par des données tactiles).

L'étude de la préhension artificielle n'en est qu'à ses débuts et de nombreux problèmes sont encore à résoudre. Cependant, le savoir-faire actuel permet déjà d'assurer la saisie automatique de pièces mécaniques prédéfinies et de replacer le geste préhenseur dans le contexte général du comportement du robot dans son environnement.

Christian Laugier, directeur de recherche à l'INRIA, Laboratoire d'informatique fondamentale et d'intelligence artificielle (LIFIA) (URA 394 CNRS), 46, avenue Félix Viallet, 38031 Grenoble Cedex.

Jocelyne Troccaz, chargée de recherche au CNRS, LIFIA (URA 394 CNRS), détachée au laboratoire TIMC, Faculté de médecine, Domaine de la Merci, 38700 La Tronche.

Main articulée équipée de capteurs tactiles pour la manipulation dextre d'objets (Salisbury, MIT).

PLANIFIER OU RÉAGIR

De quelle architecture logicielle faut-il munir un robot autonome pour qu'il accomplisse sa tâche dans une durée limitée ?

Raja Chatila

PLANNING OR REACTING - Should an autonomous robot always plan its actions beforehand or only respond at all times to its environment stimuli? A combination of decision-making capacities and controlled reflex reactions is probably the correct answer.

La planification est la faculté de formuler à l'avance la suite des actions permettant d'atteindre un objectif donné, compte tenu d'un état du monde et des moyens disponibles. C'est un processus complexe dont le temps de calcul, dans le cas général, n'est pas borné. Les capacités de planification sont liées à celles de perception et de représentation.

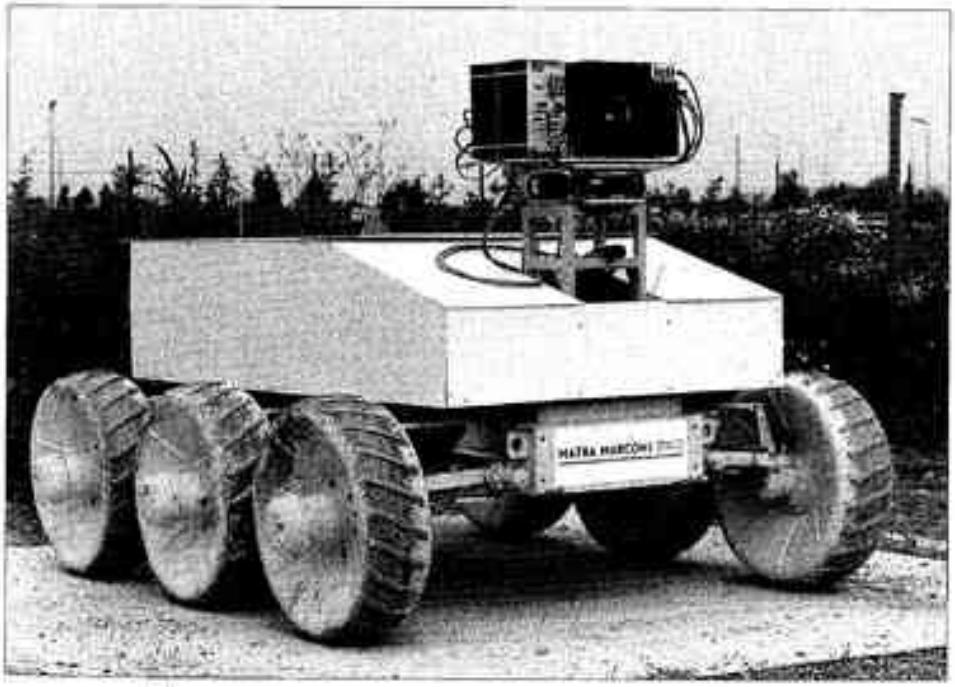
Pour un robot autonome se pose la question de la mise à jour des plans, en fonction de son évolution dans l'environnement, ou des changements de celui-ci. Mais un système qui devrait systématiquement replanifier son comportement à chaque instant serait hautement inefficace, car il passerait son temps à réfléchir plutôt qu'à agir.

A l'opposé, il est possible de concevoir un robot ne possédant aucune capacité de planification et ayant des systèmes de perception simples, pouvant se limiter à une détection de contact, de lumière. Le calcul de trajectoire pourrait être quasi inexistant, le robot errant sans but défini, ou étant directement guidé par l'intensité lumineuse perçue par exemple. Un tel robot serait purement réactif, il répondrait directement aux stimuli que sont les modifications locales de son environnement par des réactions préétablies. N'ayant qu'une connaissance locale et instantanée, il serait incapable de prédire un enchaînement d'actions tant soit peu complexes et d'avoir un comportement cohérent vis-à-vis du but à atteindre.

La recherche de la réponse appropriée

La bonne solution doit se situer entre ces deux extrêmes. Les actions sont soumises à des contraintes : instants de début et de fin, dynamique propre, synchronisation avec d'autres actions. L'environnement peut avoir sa dynamique propre, en particulier si d'autres robots y sont présents. Dans ce contexte, un robot sera dit "réactif" s'il est capable de détecter les événements pertinents et d'y fournir une réponse dans un temps compatible avec les vitesses d'évolution de l'environnement et de la tâche.

Il s'agit donc pour le robot de fournir une "réponse appropriée", c'est-à-dire correspondant à l'action désirée, mais réalisée dans un temps limité par l'évolution de l'environnement et les contraintes liées aux actions. Pour un robot réel, le compromis ne peut être obtenu que par une bonne organisation de l'architecture de ses logiciels de traitement et de commande. C'est bien, en effet, la structure décisionnelle du robot qui déterminera à la fois sa capacité à mener une tâche de façon autonome et le temps qu'il mettra pour y arriver. Diverses architectures ont été proposées pour y parvenir.



Robot mobile sous : le déplacement vers un objectif en milieu non structuré et mal connu exige des capacités de perception et d'anticipation. (© Matra Marconi Space).

Trois types d'architecture

Inspirée des modèles élaborés en éthologie, l'architecture de contrôle proposée par R. Brooks (MIT) est constituée par une hiérarchie de couches de contrôle successives, composées de processus bouclés. Chaque couche correspond à un comportement qui permet au robot d'opérer à un certain niveau de compétence (errer au hasard en évitant le contact, éviter les obstacles, aller dans une direction donnée, etc.). Un niveau donné peut inhiber et remplacer un niveau directement inférieur, qui continue toutefois à fonctionner et qui peut reprendre le contrôle dès que ▶

l'inhibition est terminée. Le comportement global du robot est le résultat de l'opération parallèle des divers niveaux qui réagissent aux données sensorielles. A cette structure qui assure une très bonne réactivité au niveau action, mais dans laquelle toute anticipation est absente, correspond aussi une implantation matérielle adaptée.

L'architecture *Task Control Architecture* développée à CMU (R. Simmons) est, à l'inverse, basée sur une structure purement centralisée. Le robot-système est décomposé en modules dont les requêtes passent par un système de contrôle central qui les interprète et les transmet aux modules concernés. Cette architecture est adaptée à un robot avec une réactivité limitée et qui ne met pas en œuvre des boucles asservies perception/action.

L'approche actuellement développée au LAAS vise à obtenir une réactivité suffisante, y compris au niveau planification. Elle consiste à concevoir une architecture en trois niveaux : planification, affinement de plan et contrôle, et un ensemble de modules intégrant les fonctions de traitement dont les relations varient selon les tâches exécutées. Cette architecture est conçue pour obtenir une réponse graduée du robot aux événements du monde. Une première réaction "réflexe", à temps de déclenchement constant, est produite grâce à des fonctions simples et programmables au sein des modules de traitement. Une réaction de correction – en temps calculable – est ensuite élaborée par un "contrôleur d'exécution" disposant d'informations sur la tâche en cours et sur son évolution à court terme. Enfin, une réaction prédictive de remise en cause, donc de replanification qui est à temps non borné, est fournie par le planificateur si nécessaire.

Ces trois types de réactivité se déroulent en parallèle, mais se passent à des échelles de temps différentes eu égard à la complexité des processus mis en œuvre à chaque niveau.

Raja Chatila, chargé de recherche au CNRS, Laboratoire d'automatique et d'analyse des systèmes (UPR 8001 CNRS), 7, avenue du Colonel Roche, 31077 Toulouse Cedex.

UN ROBOT POUR EXPLORER MARS

L'exploration des planètes par des robots pose un problème difficile. Le temps mis par les ondes électromagnétiques pour faire l'aller et retour Terre-Planète est trop long pour que le robot puisse être totalement télécommandé. Il faut donc construire un véhicule largement autonome.

C'est la raison pour laquelle le CNES a engagé au début de 1989 le programme "Véhicule Automatique Planétaire" (VAP). Un groupement "Robots d'Intervention sur Site Planétaire" (RISP) a été constitué par le CEA, le CNRS, l'INRIA et l'ONERA pour répondre, en partenariat avec le CNES, aux questions difficiles que pose ce programme. L'objectif affiché est une exploration de Mars au début du prochain millénaire.

La problématique a permis de distinguer deux grandes composantes :

- la station opérateur, comprenant des moyens puissants de traitement des informations et de maintien de leur cohérence, des fonctions de programmation, de conduite et de surveillance ;
- le robot proprement dit, avec trois grands domaines où l'apport innovant est considérable : perception et modélisation de l'environnement, structure et autonomie décisionnelles, ainsi que génération des déplacements, et enfin architecture de locomotion et manipulateur embarqué.

Les résultats de ces études se sont traduits par la définition en 1991 de l'ensemble des concepts à retenir pour les sous-systèmes.

Le CNES a souhaité au cours de la seconde phase 1991-1993 développer en priorité les systèmes constituant le robot. Cette phase a été orienté dès 1992 vers :

- la réalisation de maquettes fonctionnelles pour valider les concepts, l'algorithmique, les logiciels et l'instrumentation ;
- la conception d'un démonstrateur au sol.

Celle-ci doit se poursuivre à partir du début 1993 avec des partenaires industriels choisis par le CNES dans le cadre d'un important projet de coopération internationale sur les aspects robotique.

Le projet EUROPA qui réunit, à côté de la France, l'Espagne, la Hongrie et la Russie, aboutira à la réalisation du démonstrateur et à son expérimentation en 1995.

Georges Giralt, directeur de recherche au CNRS, président du groupement RISP, Laboratoire d'automatique et d'analyse des systèmes (UPR 8001 CNRS), 7, avenue du Colonel Roche, 31077 Toulouse Cedex.

Vue d'artiste de l'avant-projet du véhicule automatique planétaire (VAP) (Cliché CNES)

L'INTELLIGENCE ARTIFICIELLE DISTRIBUÉE

Quand le savoir est distribué entre plusieurs agents, l'intelligence artificielle change de nature : il faut parvenir à coordonner leurs actions pour obtenir un résultat collectif.

DISTRIBUTED ARTIFICIAL INTELLIGENCE
Getting different artificial intelligence systems to cooperate is in theory similar to the task of coordinating human action. As such, these investigations are based on notions comparable to those of sociology. According to their school of thought, researchers prefer either to deal with complex operators (intelligent ones) or cruder ones whose intelligence is the outcome of mutual cooperation.

Jacques Ferber

Les approches classiques de l'intelligence artificielle ont montré, en particulier avec le développement industriel des systèmes experts, que la conception d'une base de connaissances soulève de nombreuses difficultés. La plupart d'entre elles proviennent de ce que, dans une application complexe, l'expertise, le savoir-faire, les compétences, la connaissance, sont détenus par des individus différents qui, au sein d'un groupe, agissent, communiquent, échangent leurs points de vue et collaborent à la réalisation d'un but commun, même si dans le partage des tâches chacun a un but qui lui est plus personnel.

L'intelligence artificielle distribuée (IAD) est née de la nécessité de trouver des parades à ces difficultés. C'est un domaine de recherche en pleine effervescence qui devrait trouver son plein épanouissement dans les années 90. L'IAD se différencie de l'intelligence artificielle classique (IA) non seulement par le concept de distribution de l'intelligence, mais aussi par un paradigme différent. L'IA s'appuie sur une centralisation de l'intelligence au sein d'un système unique qui est conçu pour représenter un être humain dans l'accomplissement d'une tâche requérant à la fois des connaissances, de l'expérience et une certaine dose de raisonnement. L'IAD, par contre, s'appuie sur la distribution de l'intelligence entre des agents formant une société dans laquelle chacun a une certaine autonomie. Au sein de cette communauté tous travaillent, selon des modes parfois complexes de coopération, conflits, concurrence, pour aboutir à la réalisation d'un objectif global : la résolution du problème, l'établissement d'un diagnostic, la construction d'un plan, etc. L'IAD fait appel à un grand nombre de concepts nouveaux tels que la coopération, la coordination d'actions, la négociation, les conflits, la satisfaction, l'action et la réaction, la perception, etc.

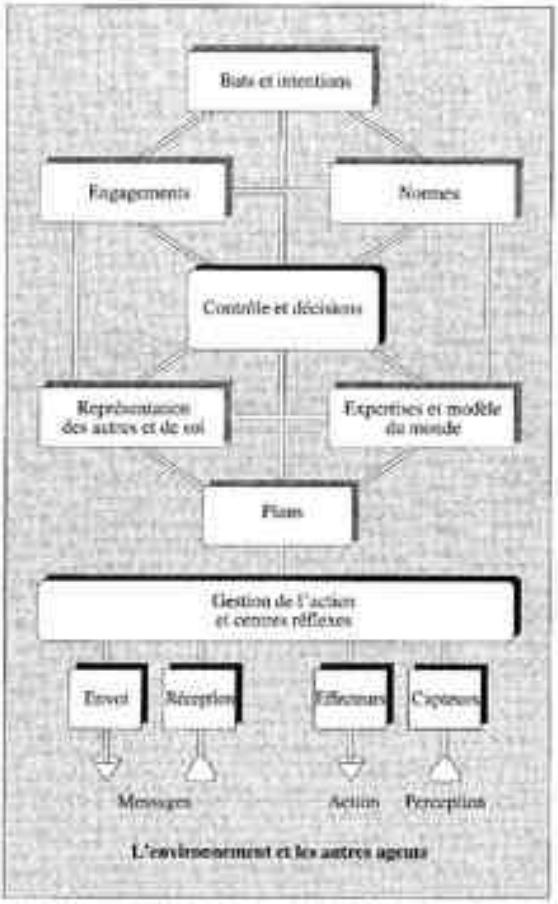
Des agents intelligents ou non ?

Faut-il concevoir les agents comme des entités déjà "intelligentes", c'est-à-dire capables de résoudre certains problèmes par eux seuls, ou bien faut-il les assimiler à de petits êtres extrêmement simples réagissant directement aux modifications de l'environnement ?

Ces deux conceptions ont donné lieu à deux écoles de pensées. La première, l'école "cognitive", est la plus représentée en IAD car elle trouve son origine dans la volonté de faire communiquer et coopérer des systèmes experts classiques. Elle considère des systèmes comprenant un petit nombre d'agents déjà complexes qui peuvent être assimilés à de véritables systèmes experts. Chaque agent dispose d'une base de connaissances comprenant les diverses informations liées à son domaine d'expertise et à la gestion des interactions avec les autres agents et son environnement. Les agents sont généralement "intentionnels", c'est-à-dire qu'ils possèdent des buts et des plans explicites leur permettant d'évoluer vers leur but. Dans ce cadre, les problèmes de coopération ressemblent étonnamment à ceux de petits groupes d'individus qui doivent coordonner leur activité et sont parfois amenés à négocier pour résoudre leurs conflits. Les analogies sont sociales, et nombre de chercheurs en IAD s'appuient sur les travaux de sociologie, et en particulier sur la sociologie des organisations et des petits groupes.

L'autre tendance, l'école "réactive", prétend au contraire qu'il n'est pas nécessaire que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent. Des mécanismes de réactions aux événements, ne prenant en compte ni une explicitation des buts, ni des mécanismes de planification, peuvent résoudre des problèmes qualifiés de complexes. Cette école est moins représentée, mais ses travaux avancent rapidement en utilisant une approche plus expérimentale que l'école cognitive à tendance plus formelle.

L'intelligence artificielle distribuée utilise tout un ensemble de concepts dont certains sont issus de la biologie ou de la sociologie. Chacun d'entre eux pris séparément est assez simple, mais leur ▶



Architecture d'un agent cognitif.

articulation pose encore de nombreux problèmes, que l'on peut classer en cinq catégories :
- les modes de communication, et les protocoles utilisés : la théorie des actes de langage considère que communiquer est une action comme une autre, qui est la conséquence d'intentions, et qui est soumise à des critères pragmatiques de condition de satisfaction. Il est donc possible de traiter formellement la communication comme un acte ;
- la représentation, les compétences des agents et l'allocation des tâches : il s'agit de représenter la connaissance que les agents ont des autres, en termes de croyances, et de répondre aux questions : qui fait quoi ? qui sait faire quoi ? qui veut faire quoi ? qui sait quoi ?... et d'utiliser cette connaissance à des fins de coopération et de répartition des tâches entre agents ;
- l'intentionnalité et sa relation à l'action : la motivation qui porte un agent à faire en sorte qu'une situation advienne, est à la base de l'autonomie des agents cognitifs. Il s'agit de relier ses buts, ses possibilités d'actions, ce qu'il sait des autres et les engagements qu'il a contractés dans une perspective globale qui lui permette d'aboutir dans ce qu'il entreprend ;
- la coopération par coordination d'actions : il s'agit de déterminer les méthodes à mettre en jeu pour que plusieurs agents devant coordonner leurs actions soient capables, ensemble, d'atteindre leurs buts ;
- la gestion des conflits et la négociation : l'autonomie des agents introduit nécessairement des conflits qui doivent être résolus. L'une des méthodes les plus employées en IAD est la négociation qui suppose que deux (ou plus) agents soient capables d'entrer dans un processus de proposition, d'évaluation et de contre-proposition jusqu'à la satisfaction des parties en présence.

Les systèmes cognitifs sont les plus complexes

Les systèmes cognitifs tendent vers une complexification de plus en plus grande. L'intention est liée aux actions par le biais des connaissances sur autrui. Chaque geste, chaque raisonnement doit prendre en compte les possibilités d'action des autres agents, et s'il y a un conflit, entrer éventuellement dans une phase de négociation. De ce fait, on ne sait gérer que des systèmes de quelques agents cognitifs, au maximum une dizaine. Lorsque le nombre augmente, soit on cherche à réunir ces agents en groupes, chaque groupe ne contenant qu'un petit nombre d'agents, soit on simplifie le travail en définissant des rôles bien particuliers de manière que le nombre d'interactions reste relativement faible.

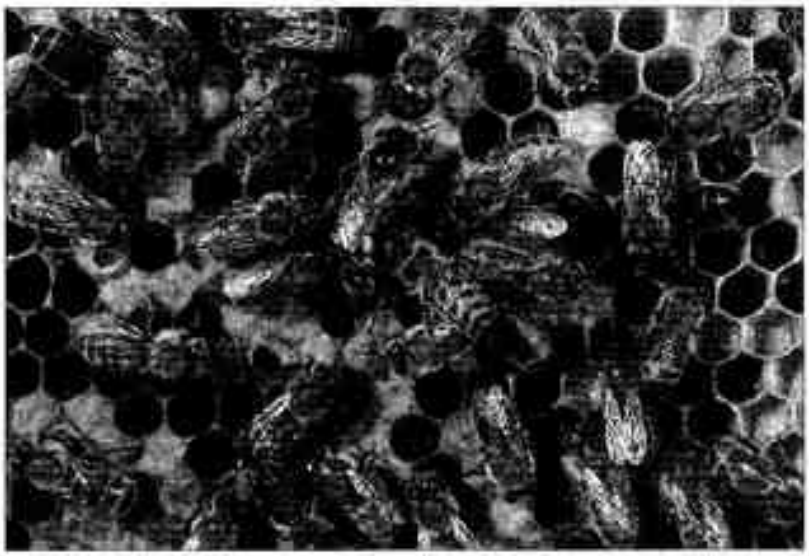
Devant cette complexité, une autre voie de recherche consiste à utiliser des agents extrêmement simples et ne disposant de presque aucune capacité de raisonnement. Chaque agent dispose d'un comportement presque "câblé" tel que se laisser attirer par un signal (une lumière, un son, ou n'importe quel rayonnement), laisser des traces dans l'environnement, etc. Ces agents interagissent toutefois, et c'est de ces interactions locales que naissent des structures qui apparaissent globalement organisées. L'intelligence émerge ainsi des interactions d'un grand nombre d'agents qui individuellement ne disposent d'aucune intelligence. L'analogie la plus utilisée dans ce domaine est celle de la fourmilière, de la termitière ou de la ruche d'abeilles : bien que chaque insecte soit relativement simple et ne dispose que d'une gamme de comportements extrêmement réduite, toute la société d'insectes fonctionne comme un tout. Tout se passe comme si elle disposait d'un grand organe centralisateur, capable de prendre les décisions, et donnant des ordres aux différents éléments de la société. Et pourtant il n'existe aucune centralisation, aucune hiérarchie de commandement. Les comportements simples de chacun des éléments de la société produisent des résultats qui deviennent des stimuli pour les autres membres. Il y a donc bien émergence d'une organisation sans plan préalable. Les travaux sur les agents réactifs ont déjà permis de montrer qu'il était possible de réaliser des tâches complexes (tri collectif, récupération d'objets disséminés, synchronisation d'actions, résolution de problèmes, ...) par simple interaction de tels agents.

Les applications de l'IAD ne se résument pas à la construction de compagnies de robots ou de systèmes multi-experts. Comprendre les processus d'évolution des systèmes complexes est l'un des défis majeurs de la biologie et des sciences sociales et leur simulation pose de nombreuses difficultés du fait de leur inhérente complexité. En modélisant les comportements des individus d'une société, et en les faisant interagir à l'intérieur de "mondes artificiels", l'IAD est capable de simuler des systèmes écologiques ou sociaux complexes et ainsi de faire émerger des structures qui résultent du passage des études micro-sociales aux analyses macro-sociales.

Les perspectives de l'IAD pour les sciences de l'homme et de la nature sont considérables, car des domaines tels que l'éthologie, la géographie ou la sociologie peuvent enfin disposer de véritables "laboratoires virtuels" dans lesquels tous les paramètres sont connus et modifiables indépendamment. Le résultat, confronté aux études de terrain, assure alors une validation d'hypothèses presque impossible à atteindre autrement.

L'intelligence artificielle distribuée constitue un domaine particulièrement prometteur bien qu'encore peu étudié en France et dont les possibilités sont encore largement à explorer.

Jacques Ferber, professeur à l'Université Pierre et Marie Curie, Laboratoire formes et intelligence artificielle (URA 1095 CNRS), Université Pierre et Marie Curie, 4, place Jussieu - Tour 46-0, 75252 Paris Cedex 05.



Un système d'intelligence artificielle distribuée peut être comparé à une ruche : un ensemble d'agents parfois très simples peuvent accomplir par coopération des tâches complexes. (© CNRS / Ecomop / A. R. Deves).

INFORMATIQUE ET MÉDECINE

Aide au diagnostic, archivage d'information et traitement d'image, la médecine est destinée à utiliser de plus en plus couramment l'informatique moderne.

Jacques Demongeot

DATA PROCESSING AND MEDECINE. - Recently, computers have been involved in assistance for diagnosis, medical follow-up of patients. Imaging is the most promising tool either directly to support medical intervention (for instance, determining a surgical path) or, by superimposition, to follow a drug's progress to its final destination in the body.

L'informatique médicale connaît de rapides développements depuis une dizaine d'années, sur un large spectre d'applications, et elle utilise presque tous les outils de l'informatique classique. Les trois grands domaines d'application sont :

- les systèmes d'information hospitaliers intégrés, permettant de mettre au point des dossiers médicaux de spécialité, spécifiques de chaque unité de soins, puis de les réunir par la mise en place d'un réseau de communication comportant un sous-réseau image et un sous-réseau serveur de résultats ;
- les systèmes d'aide à la décision médicale, dont le but est d'aider la démarche diagnostique et le choix thérapeutique ;
- les systèmes d'information image, qui autorisent aussi bien la visualisation tridimensionnelle que la communication des images à des stations de travail spécialisées, et même l'aide aux interventions médico-chirurgicales.

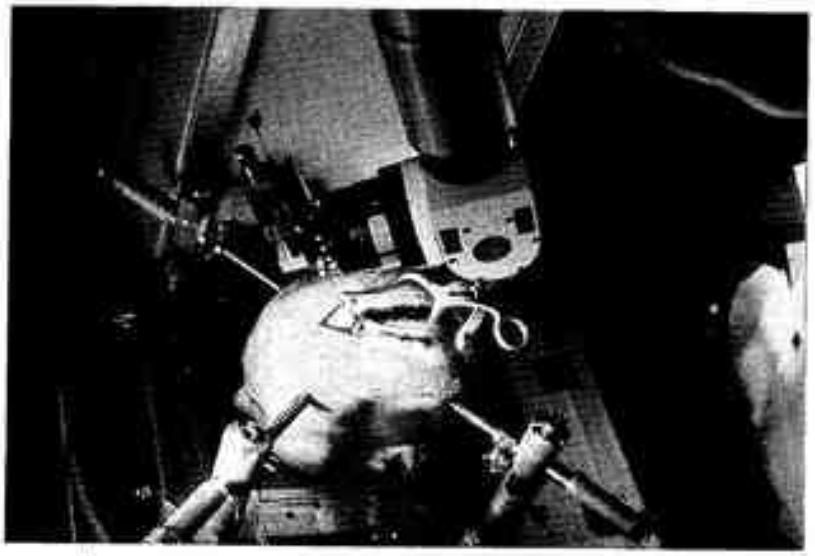
L'imagerie est l'outil principal

On distingue quatre types principaux de systèmes d'information image : l'imagerie anatomique (tomodensitométrie, IRM, angiographie numérisée,...), l'imagerie fonctionnelle (gammatomographie, magnéto-encéphalographie,...), et, en plein développement, l'imagerie positionnelle (acquisitions laser, vidéo, infrarouge, ultrasonore, électromagnétique, tactile,...) et l'imagerie encyclopédique (atlas digitalisés, CAO et imagerie de synthèse,...).

Ces systèmes d'information image posent quatre défis à la médecine :

- les gestes médicaux assistés par ordinateur représenteront une part croissante de l'activité d'une salle d'opération moderne. Les gains attendus sont un surcroît de rapidité, de précision et d'optimalité dans le choix d'une voie d'abord chirurgicale, la détermination des trajectoires et même l'exérèse. Cette technique pourra être entièrement informatisée (robot exécutant le travail du chirurgien dans les cas simples de ponctions-biopsies d'organes situés dans un environnement aux rapports anatomiques constants), partiellement informatisée (proposition d'un guide d'outils pour des opérations nécessitant une procédure manuelle d'exécution du geste), ou seulement interactive, avec un écran présentant en 3D la zone d'intérêt et la trajectoire idéale à suivre, ainsi que la trajectoire acquise durant l'opération, de manière à permettre les corrections de celle-ci en temps réel. Dans une salle d'opération de routine, on pourra ainsi disposer d'un robot fixé au sol permettant des gestes tels que la stéréotaxie cérébrale, et d'un robot pantographique fixé au plafond, sur lequel pourront être placés des outils de manière à optimiser l'angle d'observation ou d'action. Des moniteurs haute-résolution seront également disponibles pour la visualisation des objets placés derrière le plan chirurgical atteint à tout moment de l'opération, ainsi que l'extrapolation de la trajectoire d'outils tenus à la main ou par un robot, acquises en temps réel par imagerie positionnelle laser ou vidéo. On disposera pour cela des moyens d'imagerie locaux permettant le feed-back en temps réel sur la position du patient ou celle des outils ;
- l'affichage synoptique ou superposé, sur écran, d'images anatomiques et fonctionnelles (3D scintigraphique superposé à un 3D IRM de poumons par exemple) permettra la visualisation des zones pathologiques dans leur environnement anatomique ;
- la console d'imagerie du clinicien devra permettre l'affichage simultané du dossier médical du patient et des images de son dossier d'imagerie, ainsi que l'accès à des ressources d'imagerie encyclopédique : mises en correspondance avec les images réelles du patient, elles faciliteront la lecture des images logiques interprétées par l'imageur ;
- dans l'utilisation d'anticorps monoclonaux spécifiques pour la fixation d'éléments radioactifs sur certains organes malades, l'image sur console permettra de vérifier l'adéquation des zones atteintes et des zones traitées. Dans un second temps, le contrôle d'embols guidés dans les vaisseaux par un champ magnétique devrait rendre possible l'administration *in situ* de la drogue efficace. Dans les deux cas, la mise en correspondance d'une information préthérapeutique avec une information en temps réel sur la localisation de la drogue sera efficace.

Jacques Demongeot, professeur à l'Université Joseph Fourier, IMAG, TIM B, Faculté de médecine, Domaine de la Merci, 38700 La Tronche.



Robot en neurochirurgie stéréotaxique (collaboration TIMB et Service de neurochirurgie CHUG).

LA TÉLÉOPÉRATION

La télémanipulation est l'ancêtre de la téléopération qui permet aujourd'hui d'agir à distance, par l'intermédiaire d'un télérobot.

Bernard Espiau

La téléopération a son origine dans la télémanipulation, elle-même conçue comme un prolongement du geste humain vers un univers de travail dit "hostile".

Initialement destinée à répondre aux besoins de l'industrie et de la recherche nucléaire vers les années 50, la télémanipulation a accompagné l'évolution des techniques et des technologies. Elle est aussi devenue de plus en plus complexe, au point de se transformer si profondément qu'il a été nécessaire de changer son nom en téléopération. Aujourd'hui, deux grandes tendances se dégagent. L'une va vers la télérobotique, ultime étape coopérative sur le chemin de la robotique autonome, dans laquelle l'opérateur se limite à donner une mission au robot, à recevoir son compte-rendu, et éventuellement à modifier les programmes utilisés. L'autre va vers la téléprésence qui vise à une transparence quasi totale du système : l'opérateur doit pouvoir percevoir à distance l'environnement de travail et y agir comme s'il y était réellement. La notion de monde virtuel trouve là une justification particlie. Entre ces deux extrêmes, toute une gamme de nuances est jouée sur le thème de la coopération.

La programmation de la téléopération présente des caractères spécifiques : environnement non structuré ou mal connu, non répétitivité des tâches à réaliser, coût et enjeux élevés, présence de l'homme à divers niveaux, contraintes de communication (retards, faibles bandes passantes, manque de fiabilité de la transmission et de puissance de calcul embarquée). Il est par exemple courant de constater, dans les applications sous-marines ou spatiales, un retard de transmission de 2 à 8 secondes, alors qu'un retard de quelques dixièmes de seconde du retour d'effort suffit à empêcher toute commande manuelle interactive.

Programmation en temps réel et informatique distribuée

La programmation en téléopération doit donc tenir compte des impératifs de temps provenant de la transmission, de la réaction de l'opérateur et de l'évolution de la tâche, posant ainsi des problèmes de temps réel. Par ailleurs, les traitements de plus bas niveau seront plutôt menés du côté des actionneurs, alors que les décisions de plus haut niveau seront prises du côté de l'opérateur : on rencontre ainsi des problèmes d'informatique distribuée.

La téléopération implique le traitement d'ensemble d'informations symboliques et numériques, explicites ou implicites, en tenant compte des caractéristiques de l'opérateur. La représentation graphique, et plus généralement la communication multimédia, apparaît comme devant jouer un rôle important dans le futur. Mentionnons en particulier l'intérêt des représentations de scènes en trois dimensions, remises à jour en permanence, dans lesquelles s'incrustent des informations et des propositions d'action offertes à l'opérateur. Dans ce contexte s'estompe la distinction traditionnelle entre simulation (exécution hors ligne, ou différée) et exécution en ligne. Ainsi, pour compenser l'effet d'un retard de transmission, la prédiction d'un comportement peut-elle être utilisée conjointement à un contrôle réel.

Un dépannage étant par nature impossible, la fiabilité et la sûreté de fonctionnement sont des facteurs déterminants. Des méthodes efficaces de validation, par exemple basées sur une approche synchrone, s'imposent. Elles s'accompagnent de la mise en place de systèmes de détection des défaillances, avec pour objectif la possibilité de restaurer la capacité d'exécution dans tous les cas non critiques de mauvais fonctionnement.

REMOTE OPERATION - Remote operation is an extension of remote handling as practised in the 50's. It is more particularly intended for situations where transmission lags are long and or transmission is unreliable. The human operator may find it useful to view 3D scenes containing both information about the actual situation and possible courses of action.

Opération de télémanipulation en cellule blindée au centre de la Hague-AD CEA.

Bernard Espiau, directeur de recherche à l'INRIA. 2004, route des Lucioles. BP 93. 06902 Sophia-Antipolis Cedex.

LE COPILOTAGE AUTOMOBILE

Les études actuelles portent non pas sur le pilotage automatique des voitures, mais sur le copilotage c'est-à-dire une assistance automatisée à la conduite qui reste assurée par l'homme.

Bernard Dubuisson

MOTOR VEHICLE CO-PILOTING - The step preceding automated piloting is co-piloting. The ProLab 1 vehicle warns the driver when he deviates from his route, encounters an obstacle and when he can pass another vehicle. Its successor should be able to solve more complex driving situations.

De façon générale, un système d'assistance au pilotage, que nous désignerons par copilotage, doit intégrer quatre modules :
- un module de perception dont la fonction concerne l'acquisition et le traitement des données issues de différents capteurs (tachymètre, accéléromètre, télémètre, caméras, contact clignotant,...) équipant le véhicule ;
- un module de contrôle dynamique dont le rôle est de calculer des informations de sécurité, comme la distance de sécurité à l'avant du véhicule, et de déterminer, si nécessaire, les caractéristiques de dépassement de l'obstacle ;
- un module de diagnostic chargé d'évaluer la situation du véhicule afin de déterminer le niveau de danger de celui-ci. Ce module se comporte en maître vis-à-vis des autres : il leur indique leur mode de fonctionnement et peut leur demander des informations complémentaires ;
- un module chargé de la présentation des informations au conducteur. C'est via ce module que le module de diagnostic signale un danger au conducteur et lui propose une manœuvre adaptée.

Le véhicule ProLab 1

Le véhicule ProLab 1 résulte d'études menées conjointement par plusieurs laboratoires et industriels français. Son système d'aide au pilotage traite plusieurs tâches qui coopèrent entre elles :
- la tâche "suivi de lignes blanches" transmet toutes les 20 ms, par une liaison directe, à la tâche "capteurs" la position du véhicule par rapport à la ligne médiane de la chaussée ;
- la tâche "capteurs" effectue un tir télémétrique avec une période de 1 ms et un traitement visuel, après recueil des informations de la tâche *tracking*, toutes les 20 ms. Cette tâche acquiert également toutes les 20 ms la valeur des paramètres vitesse et contact clignotant ;
- la tâche *tracking* transmet toutes les 20 ms les informations mises en forme par le système de vision. Toutes les 200 ms, elle active la tâche "diagnostic" ;
- la tâche "diagnostic" a diverses fonctions. Suivant la situation du véhicule (à l'arrêt, en mouvement,...), elle active la tâche "contrôle dynamique" pour que celle-ci détermine la distance de sécurité à l'avant du véhicule et, si nécessaire, les caractéristiques de dépassement en termes de temps et de distance à un obstacle donné. Elle guide également la tâche "capteurs". La tâche de diagnostic terminée, les résultats sont transmis au module de présentation. Suivant ce schéma, la tâche diagnostic doit être en mesure d'effectuer ses traitements dans un intervalle de temps inférieur à 200 ms, de même pour la tâche "contrôle dynamique".

Le système de diagnostic met en œuvre des structures multi-experts communiquant par passation de messages avec boîte aux lettres. Le fonctionnement en temps réel est assuré par des techniques de focalisation d'attention (présélection des règles correspondant à la situation en cours) et de précompilation des réseaux de règles : le moteur d'inférence peut déclencher jusqu'à cent mille règles par seconde sur un processeur de type 486.

La réalisation du prototype ProLab 1 et son expérimentation sur un véhicule roulant à la vitesse maximale autorisée sur autoroute (130 km/h) a été la concrétisation des efforts de recherche des trois dernières années. Elle constitue la première phase du projet de recherche ProLab qui, dans la suite, doit résoudre des scénarios complexes dans le cadre de situations routières.

Tableau de bord du véhicule ProLab 1. L'écran situé à droite du volant donne des alertes au conducteur par rapport aux manœuvres envisagées.

Bernard Dubuisson, professeur à l'Université de technologie de Compiègne, directeur du Laboratoire heuristique et diagnostic des systèmes complexes (URA 817 CNRS), UTC , rue Personne de Roberval, BP 649, 60206 Compiègne Cedex.

La machine communicante

Maquette de commande vocale d'un poste de contrôle aérien avec simulation des pilotes (étude CDH-LIMSI-SEXTANT-AVIANE/CDVS, cliché LIMSI).

La communication entre l'homme et la machine est un sujet de recherche en pleine expansion. Elle constitue l'un des principaux domaines sur lesquels les fabricants d'ordinateurs, de matériels de télécommunication, ou d'électronique grand public portent leurs efforts de recherche et développement.

Communiquer, pour un humain, ce sera percevoir ou produire un message ou une action, à partir d'une opération cognitive, explicite ou implicite, sur un ensemble de connaissances. Divers moyens de communication coexistent. On y trouvera, pour la perception, les cinq sens : l'ouïe, la vue, le toucher, le goût et l'odorat. Il faut y ajouter la lecture, qui est une opération particulière de la vision, liée à la langue écrite, et la perception de la parole, cas particulier de la perception des sons.

Pour la production, on trouvera de même le support vocal, avec la production de parole ou de sons, le support visuel, avec la réalisation de dessins, de graphiques, ou d'un message écrit. Alors que ces moyens interviennent dans la communication avec les êtres humains, d'autres actions peuvent être produites (telles que saisir, jeter, soulever,...) permettant une communication généralisée à l'ensemble du monde physique.

Au point de rencontre de tous ces modes de communication, on trouve la fonction de commande et de contrôle, la cognition, permettant de comprendre ou d'engendrer le message, à partir des connaissances, dans la perspective d'un dialogue.

C'est de cette faculté, propre à l'être humain, que l'on se propose de doter la machine. Et ce faisant, autoriser un dialogue avec cette dernière. De fait, la machine intervient comme interface entre l'être humain et le monde avec lequel il communique. Ce monde peut être réduit à des objets, mais il peut également être constitué d'autres humains.

L'ordinateur possède déjà des capacités de perception artificielle, en matière de reconnaissance de parole, de sons, d'écriture, d'images ou de gestes. Il est intéressant de noter que cette fonction de reconnaissance du message transmis peut s'accompagner d'une reconnaissance de l'identité de la personne : identification du locuteur par sa voix, reconnaissance de signature, reconnaissance visuelle de l'empreinte digitale, de la géométrie de la main, du visage ou de la rétine. On notera également que reconnaissance et compréhension sont des opérations intimement liées, dans le cadre d'un processus dynamique, la compréhension d'une partie d'un message oral, écrit ou visuel intervenant sur la reconnaissance du reste de ce message. Les capacités de ces modes de communication sont encore très limitées : on place des contraintes en reconnaissance vocale (prononciation par mots isolés, respect d'un lexique réduit ou d'une syntaxe rigide, ambiance sonore calme), l'ordinateur reconnaît assez bien les caractères machines, mais bute sur les caractères manuscrits. Il est difficile de reconnaître un objet, si les conditions d'éclairage ne sont pas idéales. La reconnaissance de gestes

(communication tactilo-kinesthésique) ou de mouvement se fait par l'intermédiaire d'un gant numérique, ou même d'un habit numérique, muni de capteurs de position. Il est également possible de reconnaître la direction du regard, ou la position de la tête, grâce à des capteurs appropriés.

Réciproquement, l'ordinateur pourra produire des messages. Le plus trivial est bien sûr l'affichage sur l'écran d'un message, texte ou graphique (icones), préétabli. On y ajoutera la génération de texte à partir de concepts ou la production de résumé, la synthèse vocale, la synthèse d'images. Ces images peuvent être statiques ou animées. Elles peuvent être produites en stéréovision, être superposées à l'environnement réel, nécessitant alors le port d'un équipement adéquat. Les documents ainsi produits sont multimédias, incluant textes, images réelles ou synthétiques, statiques ou animées. De façon plus futuriste, on notera la possibilité de produire un retour d'effort dans la communication gestuelle, et de créer des formes solides simulées.

Enfin, la machine doit, elle aussi, être douée de capacités cognitives. Avoir un modèle de l'utilisateur, du monde physique sur lequel il agit, et de la communication entre ces deux protagonistes, mais également de la tâche à accomplir et des modalités du dialogue. Pouvoir mener un raisonnement, planifier une action, linguistique ou non, faire de la résolution de problèmes, de l'aide à la décision, structurer une base de connaissances multimédia, posséder une expertise permettant d'utiliser ces connaissances, fusionner les informations provenant simultanément de plusieurs capteurs, apprendre des connaissances nouvelles. La multimodalité pose le problème de la coréférence, lorsqu'on désigne par exemple un objet sur l'écran et que l'on demande oralement une information relative à cet objet ("Quel est le tonnage de ce bateau ?"). Cela entrouvre également une nouvelle problématique de recherche sur l'apprentissage de modèles de communication multimodale, par des méthodes auto-organisatrices.

Il faut souligner que la communication (production, perception et cognition) doit être optimisée globalement en agissant sur ces différentes composantes. Ainsi, l'affichage sur un écran "d' icones" que l'on peut désigner à l'aide d'une souris autorisera une communication plus efficace et plus fiable que la synthèse d'un long message parlé, suivie d'une entrée de texte au clavier. Les études relatives à l'ergonomie de ces systèmes sont donc de première importance. Il serait de même très important de pouvoir disposer de modèles de ces systèmes de communication, permettant d'en prévoir les performances et d'en améliorer le fonctionnement.

Le concept de la communication homme-machine évolue avec les nouveaux modes de communication, vision ou gant numérique, qui permettent au système d'avoir une modélisation de l'utilisateur et de ses actions. Au lieu de considérer l'utilisateur d'un côté et la machine de l'autre, l'utilisateur lui-même peut alors être partie de la modélisation de l'univers de la tâche, évoluer et agir sur cet univers, et en percevoir les réactions. La machine va donc transmettre à l'humain une représentation réelle ou modifiée du monde physique, ou produire un monde physique reconstruit (réalité virtuelle) voire de fiction.

Si le monde physique comprend lui-même des humains, le modèle devient plus complexe. La machine va alors s'intercaler entre deux, ou même plusieurs interlocuteurs, qui partagent un même référent (connaissances communes relatives au monde physique, modèles des interlocuteurs et de leur interaction). La machine pourra simplement transmettre la communication, en modifier le contenu, ou même agir sur le référent commun, permettant ainsi de créer un monde virtuel partagé.

De nombreuses applications voient le jour dans des domaines très divers, mais qui représentent un nombre de systèmes encore assez faible et ne concernent la plupart du temps qu'un mode de communication. Les applications téléphoniques du traitement automatique de la parole, reconnaissance, synthèse et codage, connaissent une forte progression. Les secteurs de l'enseignement, de la conception ou de la publication assistés par ordinateur (EAO, CAO et PAO), de la productique, du médical, de l'interrogation de bases de connaissances, sont des domaines d'application très porteurs qui devraient bénéficier des progrès réalisés en communication homme-machine multimodale.

Joseph-Jean Mariani, directeur de recherche au CNRS, directeur du Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur.

LE TRAITEMENT AUTOMATIQUE DE LA LANGUE ÉCRITE

En raison de sa complexité, le langage naturel n'est pas facile à utiliser en informatique. Bien que les besoins soient énormes, les efforts de recherche comme les résultats restent donc limités.

AUTOMATIC PROCESSING OF WRITTEN LANGUAGE - Although there is a major need in this area, automatic processing of written language is still in the Stone Age. The ambiguities, the need to be familiar with the context are obstacles that cannot be avoided. Hence, there are still very few applications of this type.

Patrick Saint-Dizier

Jusqu'au milieu des années 70, la communication avec l'ordinateur était le fait d'un petit groupe de personnes initiées au langage abstrait de l'informatique. Avec le développement des applications grand public, l'utilisation du langage naturel s'est imposée. Les premiers objectifs furent les interrogations de bases de données. Ils se sont depuis multipliés. Le traitement du langage naturel n'a plus été perçu seulement comme un moyen d'améliorer la communication homme-machine, il est devenu un objet en soi, avec ses applications propres. C'est ainsi que se sont développés la traduction automatique d'une langue dans une autre, l'indexation automatique de documents, ou l'enseignement des langues assisté par ordinateur.

La communication avec la machine en langage naturel n'est pas une panacée. Pour interroger une base de données, un mode graphique où l'usager clique sur les relations et leurs attributs peut être plus rapide que la dactylographie des mots correspondants sur un clavier. Par contre, la réponse, surtout si elle revêt un caractère intentionnel ou coopératif, sera préférablement donnée en langage naturel.

Les applications industrielles du langage naturel sont encore très peu nombreuses. Les études de marché montrent pourtant que les besoins sont énormes : quelle société ne souhaite pas automatiser la consultation "intelligente" de sa documentation technique ou bien faire traduire dans une autre langue, le plus rapidement possible, tel ou tel document ?

Ambiguïtés et données implicites

Le traitement automatique du langage naturel n'en est, en effet, qu'à l'âge de pierre. La langue est d'une complexité telle qu'elle est difficile à manipuler. Les ambiguïtés et les données implicites en sont une illustration.

Considérons la phrase :

Jean vit l'homme dans le jardin avec un télescope.

Il est impossible, en partant de cette assertion, de répondre en toute certitude à des questions telles que :

Où est l'homme ? Où est Jean ?

Qui a le télescope ? etc.

La présence d'informations incomplètes rend la compréhension difficile pour une machine. Dans le texte suivant, les mots en majuscules n'ont de sens que pour une personne qui connaît le contexte et sait, par exemple, que M. Major est le Premier ministre de Grande-Bretagne et leader du parti conservateur.

"M. Major est arrivé aujourd'hui en France. LE PREMIER MINISTRE rencontrera LE PRÉSIDENT demain. LE LEADER DU PARTI CONSERVATEUR ira ensuite à Moscou où il rencontrera M. Gorbatchev. Mme Major rejoindra SON ÉPOUX en Russie, où ce FILS D'ARTISTES DE CIRQUE est un personnage relativement peu connu".

L'objectif d'EUROTRA, projet européen de recherche et de développement, est de réaliser un système de traduction automatique entre les neuf langues officielles de la CEE. La première phase s'est achevée en décembre 1990, elle aura mobilisé environ 200 chercheurs du milieu universitaire. (© Ph. Plailly - Eurélios).

L'ordinateur peut-il comprendre ?

D'autres limites sont de nature strictement informatique. Si certaines tâches comme le traitement morphologique (qui utilise des règles déjà complexes pour tenir compte de la conjugaison des verbes, des pluriels, etc.) et syntaxique (qui concerne l'ordre et la fonction des mots dans la phrase) sont relativement bien maîtrisées, d'autres, notamment au niveau sémantique (de signification des mots et des expressions), le sont beaucoup moins. Les futurs développements auront probablement lieu à un niveau pragmatique, très complexe, où les phrases seront placées dans leur contexte avant d'être comprises.

Le traitement automatique du langage naturel est relativement peu développé en France. Si plusieurs sociétés importantes s'y intéressent (telles que ERLI, CAP-Gemini), il ne constitue pas une activité centrale et reste le plus souvent "en tâche de fond".

Pour constituer une masse critique suffisante, différents laboratoires (CNRS, Universités, CNET...) se sont structurés depuis huit ans autour de quelques thèmes fédérateurs : lexiques-syntaxiques et sémantiques, grammaire, dialogue et sémantique du discours. Leur organisation porte le nom de Groupe de Recherche Concerté Communication homme-machine.

Patrick Saint-Dizier, chargé de recherche au CNRS, responsable du laboratoire Langage naturel et programmation en logique, Institut de recherche en informatique de Toulouse (URA 1399 CNRS), Université Paul Sabatier, 118, route de Narbonne, 31062 Toulouse Cedex.

LA TRADUCTION AUTOMATIQUE

L'idée d'utiliser l'ordinateur pour traduire un texte est l'une des premières applications envisagées, dès les années 50, au début de la recherche en informatique. Si le problème paraît simple quand on reste au niveau du mot à mot, représenter correctement un message écrit dans une langue par un message écrit dans une autre langue soulève des difficultés énormes que les limites des systèmes offerts commercialement démontrent tous les jours.

Aussi la traduction par ordinateur, où tout le travail est fait par la machine, est-elle souvent remplacée par la traduction assistée par ordinateur où la machine n'est qu'une aide pour un traducteur humain.

La nécessité d'une relecture et d'une correction du texte produit n'est pas l'obstacle majeur, car on sait que toute bonne traduction humaine requiert elle aussi une relecture et passe par au moins une étape de correction, sinon deux. De plus, l'utilité principale de la traduction automatique ne réside pas dans la production de textes stylistiquement satisfaisants et directement publiables, mais dans la standardisation de textes techniques, et dans l'allégement des tâches répétitives qu'elle peut apporter aux traducteurs.

Le principal problème reste toujours celui de l'ambiguïté inhérente au langage dit "naturel", et des connaissances nécessaires pour la lever. On continue à se poser la question de l'utilisation de connaissances extra-linguistiques pour aider à cette tâche, et on cherche toujours des méthodes permettant de délimiter le contexte afin de supprimer l'ambiguïté. Faut-il imposer des contraintes au contenu du message ou à la langue même ?

Les progrès accomplis depuis une quinzaine d'années, en particulier en ce qui concerne l'analyse syntaxique et la délimitation des différents niveaux d'analyse linguistique, sont à mettre principalement au crédit de la recherche en linguistique théorique et formelle (les grammaires d'unification en sont un exemple).

Cependant, on assiste encore à des débats de fond sur la question de la validité et de l'universalité des représentations sémantiques : peut-on vraiment avoir des représentations sémantiques qui puissent servir de "pivot" entre toutes les langues, ou doit-on admettre qu'il faille créer des représentations intermédiaires différentes pour chaque paire de langues ?

Avec les progrès spectaculaires des techniques informatiques de ces dernières années, les outils accessibles aux chercheurs et développeurs permettent d'envisager une bien meilleure qualité des systèmes. Si la tendance actuelle favorise le développement d'outils d'aide à la traduction (poste de travail du traducteur) avec intégration de bases de données terminologiques et d'aides à la rédaction, d'autres directions de recherche sont à noter. Les nouvelles approches basées sur les méthodes statistiques sont prometteuses, bien que l'on sache que leurs capacités ne pourront pas dépasser un certain seuil. Par ailleurs, l'évaluation systématique des résultats est maintenant reconnue nécessaire et donne lieu à des recherches méthodologiques.

Dominique Estival, chercheur à l'Institut Dalle Molle pour les études sémantiques et cognitives, Université de Genève, 54, route des Acacias, CH-1227 Genève.

LA RECONNAISSANCE VOCALE

Parler à l'ordinateur est un rêve d'informaticien. A condition de limiter son ambition sur le nombre de locutions, le nombre de mots et le taux de reconnaissance, cela est devenu possible. Quelques systèmes sont déjà commercialisés.

Guy Pérennou

Compte tenu de la complexité du message vocal, aucun système de reconnaissance actuel ne peut traiter la parole spontanée dans n'importe quelle situation de communication. Tous imposent des contraintes d'utilisation : prononciation en mots isolés ou, au contraire, en parole continue (à travers le téléphone ou non), utilisation mono-, multi-locuteur ou indépendante du locuteur, limites sur l'étendue du vocabulaire ou sur la grammaire...

La spécificité d'un système de reconnaissance est déterminée par ces contraintes. Leur rôle est de limiter la variabilité et la complexité du message vocal dont les sources sont essentiellement :

- l'environnement sonore et les conditions d'enregistrement,
- les variations inter-locuteurs,
- les variations intra-locuteur,
- les variations contextuelles,
- la complexité linguistique.

Le traitement du signal et la reconnaissance sont en effet plus complexes en présence de bruits ou de distorsions. L'ambiance sonore et les bruits électroniques peuvent s'ajouter au signal vocal, des distorsions diverses sont introduites par le microphone, par le téléphone ou, dans certaines salles, par la réverbération.

Chacun d'entre nous parle différemment des autres : prosodie, prononciation et timbres sont caractéristiques non seulement du milieu socio-culturel mais aussi de l'individu. Pour une même personne, l'état physiologique et psychologique modifie la voix.

Le même mot peut être prononcé par une même personne, sans modification de son état psychologique ou physiologique, d'une manière qui dépend du contexte. L'initiale et la finale du mot sont les plus variables.

Il peut s'agir d'ajustements, d'anticipations ou de retards dans le positionnement des organes articulatoires destinés à faciliter la co-articulation de la fin d'un mot avec le début du suivant ou encore d'éventuelles liaisons, élisions ou dénasalisations.

Ces variations phonétiques contextuelles sont négligeables pour les mots prononcés isolément, mais elles s'amplifient pour les mots enchaînés, d'où les difficultés particulières de la reconnaissance de la parole continue et encore plus de la parole spontanée. S'y ajoutent bien entendu, dans ces derniers cas, les problèmes délicats de la séparation des mots.

VOICE RECOGNITION - Whether voice recognition uses dynamic matching or the Markovian model, whether or not it uses neural networks, its possibilities are still limited. Most systems require a learning period that is often long and strenuous. However, if its limitations and constraints are accepted, it can serve some very useful purposes, particularly for the disabled.

Les systèmes de reconnaissance vocale utilisent des lexiques et des grammaires. Plus le langage est riche et complexe, plus la reconnaissance est difficile. Un vocabulaire de quelques dizaines de milliers de mots est considéré comme grand. Il devient très grand quand il atteint cinquante mille mots. Ces nombres qui paraissent élevés ne le sont pas en réalité, car il faut compter parmi eux toutes les formes fléchies (c'est-à-dire accordées ou conjuguées) particulièrement nombreuses en français, plus limitées en anglais.

Comparaison dynamique et modèle markovien

Deux grandes méthodes de reconnaissance se sont successivement imposées.

La comparaison dynamique qui compare de manière optimale deux formes spectrales présentant l'une par rapport à l'autre des distorsions temporelles. La méthode prévoit une phase d'apprentissage où le locuteur prononce les mots du vocabulaire qui sont associés à leur représentation orthographique, et une phase de reconnaissance où le système affiche l'orthographe des mots retenus.

Le modèle markovien HMM (*Hidden Markov Model*) plus complexe est utilisé dans les situations plus difficiles (vocabulaire étendu, grand nombre de locuteurs, environnement bruité). Un modèle HMM est régi par des lois de probabilité de transition entre états successifs et d'émission de segments de signal pour chaque état. Il offre un cadre approprié de représentation des connaissances relatives à l'organisation du message vocal en unités phonétiques, en mots ou en phrases. Pour être utilisé en reconnaissance de la parole, il faut lui associer un algorithme de décodage qui fait correspondre au signal une succession optimale d'états.

Une phase d'apprentissage à partir d'un corpus d'exemples de prononciation est nécessaire pour estimer les différentes lois de probabilité qui régissent les transitions et les émissions.

Actuellement de nombreuses recherches tentent d'utiliser les réseaux de neurones qui se sont développés au cours de la décennie 1980-90 et qui sont capables d'apprentissages.

Les réseaux de Kohonen ont montré leur efficacité pour l'extraction d'informations phonétiques pertinentes. Ils peuvent être utilisés en combinaison avec les HMM. Les réseaux de type Perceptron sont utilisés pour la reconnaissance proprement dite.

La modélisation du temps dans les réseaux de neurones a été introduite dans les TDNN (*Time Delay Neural Network*). Des expériences de laboratoire ont permis d'obtenir des résultats comparables à ceux des HMM.

Les résultats actuels

La reconnaissance vocale des dix chiffres ou des lettres de l'alphabet a fait l'objet de recherches intenses. Elles permettent d'équiper les cabines téléphoniques d'un clavier vocal ou d'épeler un mot à un ordinateur, soit pour permettre à un handicapé de le programmer, soit pour corriger vocalement les erreurs d'un système de reconnaissance vocal.

Le système développé par ATT peut reconnaître, de manière indépendante du locuteur, des chiffres prononcés continuement à travers le réseau téléphonique avec un taux de reconnaissance exacte de 98,6 %. Des résultats du même ordre sont observés pour les lettres de l'alphabet.

Le programme américain DARPA (*Defense Agency Research*) a mis en concurrence les laboratoires pour la reconnaissance en parole continue d'un vocabulaire de 1 000 mots... Le résultat de l'évaluation de ces systèmes a été publié en 1990. Le système BYBLOS de BBN et celui de Lincoln Laboratory du MIT ont obtenu les meilleures performances en fonctionnement monolocuteur : moins de 2 % d'erreurs sur la reconnaissance des mots. Le système SPHINX développé à Carnegie Mellon University, l'un des meilleurs systèmes actuels, a obtenu moins de 4,5 % d'erreurs en fonctionnement indépendant du locuteur (pour des locuteurs masculins). Pour les vocabulaires plus étendus, il faut passer à la dictée en mots isolés.

Les machines à dicter monolocuteur de première génération, TANGORA 5000 et Dragon Writer 5000, admettaient déjà en 1985 des vocabulaires de 5 000 mots (il s'agit ici de formes fléchies et non d'entrées lexicales).

Pour la seconde génération actuelle, les vocabulaires atteignent plusieurs dizaines de milliers de formes de mots comme c'est maintenant le cas pour TANGORA 20000 (sortie récente de "speech server" d'IBM) et Dragon Dictate.

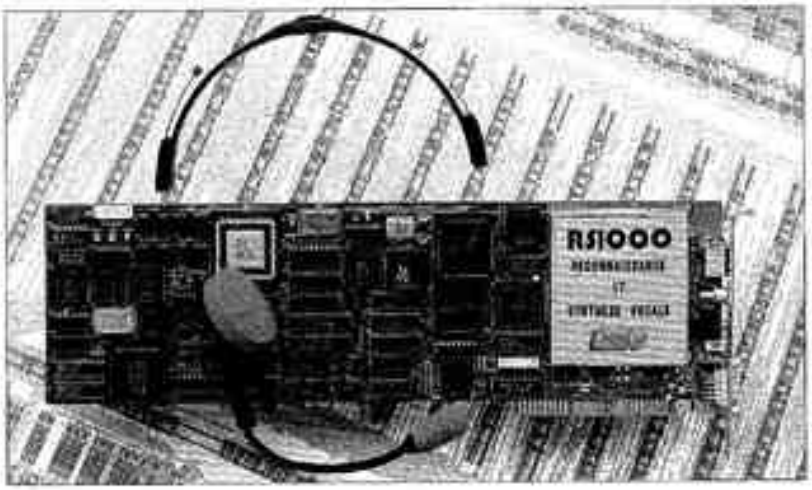
Ces machines ont un mode de fonctionnement indépendant du locuteur avec possibilité d'adaptation. Les performances médiocres au début (pour fixer les idées, disons un taux de reconnaissance de 50 %), s'améliorent progressivement. Il faut pour cela indiquer à la machine les fautes observées en cours d'utilisation de manière qu'elle adapte ses lois de probabilité. Les performances croissent progressivement pour atteindre un taux de l'ordre de 95 %.

Les contraintes d'utilisation et les prix actuels des machines à dicter en limitent la diffusion. Dragon Writer est vendu 9 000 $, une version réduite à 5 000 mots est commercialisée par IBM au prix de 3 000 $.

Selon la société Dragon, la moitié des systèmes sont actuellement achetés pour des handicapés.

Mesurer les performances

L'évaluation des performances des systèmes de reconnaissance de la parole est à la fois cruciale et délicate. La méthode la plus courante consiste à utiliser d'importants corpus de parole enregistrée permettant dans un premier temps l'apprentissage, dans un second temps les tests de reconnaissance. L'une des premières bases de données de parole enregistrée a été créée à partir de 1981 par la France dans le cadre du Pôle Parole du GDR-PRC Communication homme-machine qui a développé BDSONS (corpus de parole enregistrée) et BDLEX (lexique de l'écrit et de l'oral).



Carte d'interface vocale, développée par la société VECSYS à partir des travaux du LIMSI/CNRS. En mode monolocuteur, cette carte permet de reconnaître des phrases construites à partir de plusieurs centaines de mots prononcés en continu (fiche VECSYS).

Guy Pérennou, professeur à l'Université Paul Sabatier, Institut de recherche en informatique de Toulouse (URA 1399 CNRS), 118, route de Narbonne, 31062 Toulouse Cedex.

LA SYNTHÈSE DE LA PAROLE

Dès aujourd'hui, la synthèse de la parole permet au grand public d'avoir accès par téléphone à des informations écrites. Les systèmes de synthèse du futur, nécessairement multilingues, devront en outre pouvoir produire des types de voix très variés et s'interfacer intelligemment avec des systèmes de génération de texte.

TEXT-TO-SPEECH SYNTHESIS – Speech synthesis already allows oral access to written data-bases in operational telephone services in France. The future no doubt belongs to multilingual systems, able to speak with different voices, that could be interfaced with text generation systems towards a new generation of voice response systems for spoken man-machine dialogues.

Christel Sorin

Tout système de synthèse de parole à partir du texte procède en deux étapes : les traitements linguistiques décident des sons à émettre (phonèmes et prosodie) et les traitements acoustiques (le synthétiseur proprement dit) convertissent ces commandes en un signal de parole numérisé.

Les traitements linguistiques

Pour bon nombre d'applications, une première étape de "pré-traitement" des textes à synthétiser est indispensable : filtrage de parties imprononçables (tableaux, graphiques etc.), réécriture en toutes lettres des nombres, abréviations, sigles, restauration d'accents graphiques manquants (textes écrits en majuscules) etc. Puis vient la transcription orthographique-phonétique qui nécessite une analyse grammaticale exhaustive, ne serait-ce que pour effectuer correctement les liaisons ou traiter les mots qui s'écrivent de la même manière mais se prononcent différemment (par exemple : le couvent/elles couvent). Décider ensuite de la prosodie à appliquer à l'énoncé requiert de savoir le décomposer en groupes de mots reflétant son organisation syntaxique. Tous les systèmes de synthèse actuels possèdent donc des modules d'analyse linguistique assez performants pouvant s'appliquer à n'importe quel type de texte. Ces modules combinent la consultation de lexiques de plus en plus volumineux (10 000 à 100 000 entrées) et l'utilisation de règles de réécriture pour la phonétisation, heuristiques ou statistiques pour le découpage syntaxico-prosodique. Ils ont permis des améliorations nettes, ces dernières années, de la prosodie de la parole de synthèse. Cependant, des progrès restent à accomplir, en particulier :

- dans la spécification de grammaires prosodiques plus complètes permettant non seulement une lecture non monotone de textes longs, mais surtout la production d'énoncés "convaincants" dans un contexte de dialogue oral homme-machine (prise en compte des notions de focus, thème/rhème etc.) ;
- dans l'élaboration d'analyseurs de texte plus performants, incluant des composantes sémantiques voire pragmatiques (une machine peut-elle lire de façon "naturelle" sans "comprendre" ce qu'elle lit ?...).

A cet égard, un couplage efficace entre générateurs automatiques de texte et systèmes de synthèse de parole devrait permettre de disposer, dans un avenir proche, d'une nouvelle génération de systèmes de "réponse vocale" capables de concurrencer avantageusement le "stockage/restitution" de parole naturelle utilisé à ce jour dans les services vocaux interactifs.

Les traitements acoustiques

L'utilisation récente d'une nouvelle technique de traitement numérique du signal (PSOLA- "addition et recouvrement de signaux élémentaires") permet aujourd'hui aux synthétiseurs par concaténation d'unités préstockées (synthèse par diphones ou polyphones) de fournir un timbre de voix de synthèse de qualité identique voire supérieure à celui fourni par les synthétiseurs basés sur une modélisation de l'appareil vocal et règles acoustico-phonétiques associées (synthèse par formants ou par règles).

Si les meilleurs synthétiseurs actuels fournissent une parole très intelligible, au timbre quasi naturel, la parole de synthèse reste encore souvent perçue comme manquant de fluidité. Une des raisons essentielle est l'absence de prise en compte, au niveau des synthétiseurs, de la variabilité acoustique (segmentale et prosodique) intrinsèque à la parole naturelle. La modélisation de cette variabilité, sans doute une des conditions-clés de progrès significatifs dans le domaine, est également indispensable pour pouvoir produire différents types de voix.

La synthèse multilingue

L'avenir appartient certainement à des systèmes capables de "parler" plusieurs langues. Si les traitements linguistiques dépendent nécessairement de la langue, les synthétiseurs peuvent être rendus en grande partie indépendants de la langue (c'est le cas des synthétiseurs "par concaténation"), ce qui peut leur conférer un avantage industriel important, dans un contexte européen nécessairement multilingue.

Christel Sorin, responsable du Département recherche en communication par la parole, CNET-France-Télécom, 2, route de Trégastel - BP 40, 22301 Lannion Cedex.

PERCEPTION VISUELLE PAR MODÉLISATION GÉOMÉTRIQUE ET SYNTHÈSE

Il est dès maintenant possible d'imaginer un robot domestique qui exécute des tâches ménagères. Mais encore faut-il lui apprendre à voir !

André Gagalowicz

VISUAL PERCEPTION THROUGH GEOMETRIC MODELLING AND SYNTHESIS - If you want a domestic robot to be versatile and perform various tasks in an office or flat, it must be capable of seeing and understanding what it sees. That means it must be trained, i.e have in memory the models for the various entities in its environment. How to convey such models to it (the learning process) and how it should use them to apprehend the situation it observes are two issues dealt with here.

Nombreux sont ceux qui rêvent d'un robot domestique qui pourrait nous seconder dans de nombreuses tâches ménagères et avec lequel on pourrait dialoguer comme avec un être humain. L'analyse et la synthèse de la parole font de grands progrès et il semble possible aujourd'hui que ce robot puisse nous comprendre et nous parler dans un avenir proche. De même, les progrès de la robotique sont tels que des robots sont déjà capables d'agir, d'exécuter des tâches très minutieuses. Le problème le plus important à résoudre est de comprendre la scène dans laquelle il évolue, et localiser les objets qu'il aura à manipuler ; ce problème est très difficile dans le cas d'appartements, de bureaux où il y a un agencement complexe de nombreux objets différents. Pour résoudre ce problème, nous proposons une stratégie qui cherche à approcher celle qui est réalisée par notre cerveau.

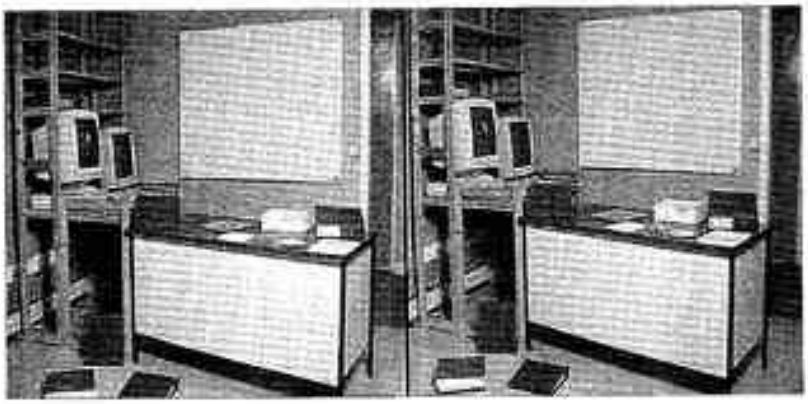


Fig. 1 : Paire stéréo d'une scène de bureau obtenue par deux caméras couleur placées sur un socle horizontal (Cliché A. Edelman, INRIA).

Remplacer les yeux

Le sens de la vue est primordial chez l'être humain. Il est assuré par le capteur visuel que sont les yeux. L'équivalent proposé est un choix de deux caméras couleur placées sur une base horizontale et montées sur le robot. Nous pouvons donc offrir à ce robot deux images couleur stéréo de la scène observée.

La figure 1 présente une paire stéréo d'images obtenues par le capteur (Laboratoire Syntim de l'INRIA) observant un bureau. Comment simuler l'activité du cerveau qui reçoit ces deux images des yeux et en déduit l'interprétation de la scène ? Pour ce faire, il faut apprendre au robot la connaissance dont il aura besoin pour comprendre la scène.

L'être humain apprend dans son enfance, et au-delà, à connaître les objets de son environnement en les observant, mais aussi en les sentant, les manipulant etc.

Cette connaissance est indispensable à son processus de reconnaissance. Malheureusement, aujourd'hui nous n'avons toujours pas l'explication de la manière dont cette connaissance est stockée dans notre cerveau, et de quelle information exacte il s'agit !

La phase d'apprentissage

Nous proposons au robot (ou à celui qui va l'utiliser chez lui) une technique pour acquérir cette connaissance indispensable qui, bien que ce ne soit pas à coup sûr la solution "humaine", semble tout à fait réalisable par celui-ci. L'idée est de livrer avec le robot un logiciel (programme) d'apprentissage, en même temps que vous recevriez ce robot chez vous. Il vous faudra exécuter ce programme interactif qui permettra de construire la base

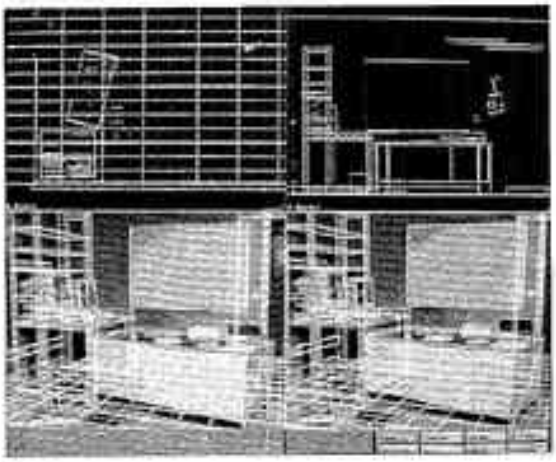


Fig. 2 : acquisition de la base de connaissances de l'espace de travail du robot : le programme d'apprentissage fournit ici un ensemble de descriptions polyédriques pour chacun des objets de l'espace de travail, que l'on peut nommer par leurs noms. La partie inférieure montre la superposition des deux images et du résultat de la construction tridimensionnelle. Les images du haut montrent les vues du dessus et de face de la base de données polyédrique (Cliché A. Edelman, INRIA).

de données géométriques représentant votre appartement. Cet apprentissage pourra vous demander plusieurs jours de travail, mais dès que le robot aura enregistré dans sa mémoire le modèle géométrique de votre appartement, le travail d'apprentissage sera terminé et votre robot sera ensuite capable, grâce à la technique décrite plus loin, d'en reconnaître tous les objets et donc d'interagir avec eux.

Nous développons à l'heure actuelle ce logiciel d'apprentissage. Il est exécuté sur une machine graphique de type mix de chez Silicon Graphics, en utilisant la souris et la console de visualisation de l'ordinateur. Grâce aux caméras du robot placé dans une première position, on visualise les deux images sur l'écran de contrôle de l'ordinateur. On fait ensuite appel à des primitives du programme (parallélépipèdes, cylindres, cônes, sphères, mais aussi surfaces splines), que l'on déplace et déforme à sa guise jusqu'à ce qu'elles viennent se projeter parfaitement sur les deux projections (gauche-droite) de la partie de l'objet que l'on veut construire. Quand la correspondance est jugée visuellement satisfaisante pour l'utilisateur, le programme restitue la forme tridimensionnelle exacte de cet objet et on passe à un objet suivant. Ce travail est un peu fastidieux car il faut analyser tous les objets un par un, mais il ne présente pas, apparemment, de difficulté d'utilisation insurmontable pour les non informaticiens. Quand tous les objets présents sur les deux images de la première position sont reconstruits, on passe à une autre partie de la scène en déplaçant la caméra.

La figure 2 présente le résultat de la reconstruction d'un tel modèle du bureau visualisé sur la figure 1. On voit la bonne juxtaposition des primitives sur les deux images du dessous. Les deux images du haut représentent, à gauche, une vue du dessus du modèle seul, et, à droite, sa vue de face. Ce sont ces données géométriques tridimensionnelles qui sont la connaissance enregistrée par le robot.

Comprendre, interpréter la scène observée

La stratégie que nous proposons pour résoudre le problème de la vision par ordinateur consiste à disposer d'une base de données complète de l'espace de travail du robot, obtenue suivant la technique décrite précédemment. Etant donné une image du type de la figure 1, ce problème revient à retrouver quelle est la partie de la base de données qui est effectivement en face du robot et quelle est la position du robot par rapport à la scène.

Bien que ce problème soit plus simple que celui qui est abordé habituellement en vision, cela reste un problème difficile. La stratégie* utilisée consiste à segmenter les deux images de la paire stéréo, puis à mettre en correspondance de manière automatique une région de l'image gauche avec la région de l'image droite qui correspond à la même facette de l'objet 3D vu dans les deux images. Ensuite, un algorithme stéréo calcule l'équation de la facette 3D en question dans l'espace. L'ensemble des facettes construites dans la phase d'analyse est ensuite comparé avec l'ensemble des facettes du modèle du monde obtenu par apprentissage. L'algorithme de reconnaissance choisit parmi toutes les possibilités celle qu'il pense être la bonne. Cet appariement optimal facettes-caméra-facettes du modèle du monde permet le calcul de la position et de l'orientation de la caméra dans le monde (l'appartement plus concrètement), ce qui permet de produire la partie de la base de données que le robot croit observer. L'idée est ensuite de calculer toute la photométrie de la partie de la scène observée par la caméra et d'en déduire, grâce aux techniques de l'infographie, des images de synthèse de cette partie de la scène. Il faut finalement comparer images de départ et images de synthèse, de manière à vérifier si l'interprétation de la scène est correcte et modifier l'interprétation et/ou la modélisation photométrique de la scène le cas échéant.

Cette technique qui fait collaborer vision par ordinateur et infographie dans une boucle de contrôle, nous semble particulièrement prometteuse. La figure 3a et la figure 3b représentent respectivement la vue gauche naturelle (à gauche) et de synthèse (à droite) et la vue droite, en utilisant pour la synthèse un simple modèle photométrique lambertien, avec une seule source de lumière située à l'infini. Le lecteur pourra constater que l'interprétation géométrique est correcte (les bons objets sont placés aux bons endroits), par contre l'aspect des images de synthèse (les images de droite de la figure 3) semble irréaliste : la texture et la réflectance des surfaces sont mauvaises et il n'y a pas d'ombres. Nous travaillons actuellement à l'amélioration de ces résultats.

En conclusion, les résultats présentés démontrent qu'il est dès aujourd'hui concevable d'imaginer l'existence future d'un robot domestique polyvalent qui sera capable de comprendre ce qu'il voit et d'agir dans cet environnement. L'évolution de la recherche dans cette direction semble aujourd'hui rendre ce rêve réalisable et est particulièrement passionnante.

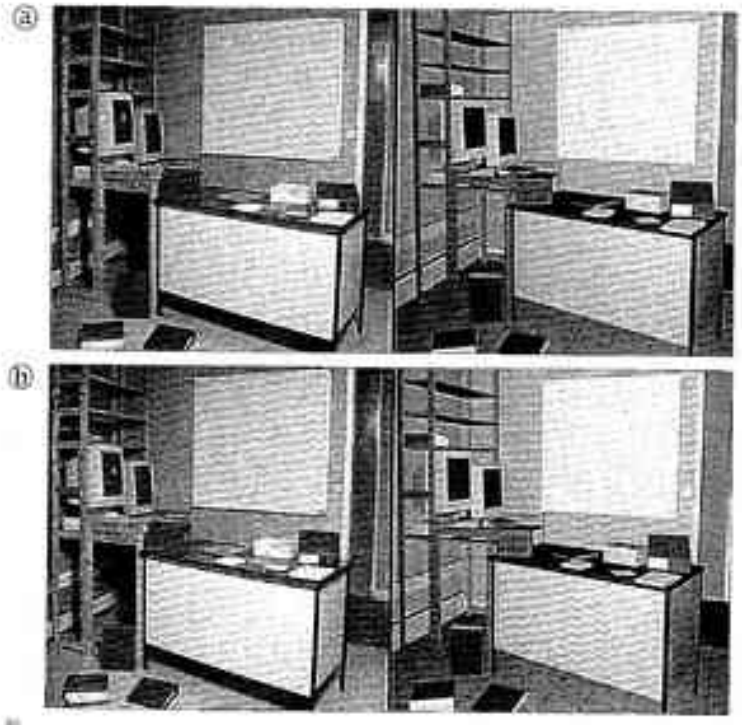


Fig. 3 : a) vues gauches, naturelle (à gauche) et reconstituée par synthèse après interprétation (à droite), du bureau de la figure 1.
b) vues droites, naturelle et de synthèse, du même bureau de la figure 1 (Cliché A. Faisman, INRIA).

* Voir l'article "L'infographie au service de la vision par ordinateur", Le Courrier du CNRS n° 77, juin 1991, du même auteur.

André Gagalowicz, directeur de recherche à l'INRIA, Domaine de Voluceau, Rocquencourt, 78153 Le Chesnay Cedex.

LA SYNTHÈSE D'IMAGES

Simuler de manière réaliste la croissance d'un arbre ou un mouvement complexe est désormais possible. L'augmentation attendue de la puissance du matériel facilitera considérablement la synthèse des images élaborées.

IMAGE SYNTHESIS - The three key words regarding image synthesis are complexity, efficiency and interaction. Progress in modelling complex objects and simulation of movement on the basis of physical laws depend on the development not only of specific techniques but also on advances in information technology generally, especially the power of the hardware available.

Claude Puech

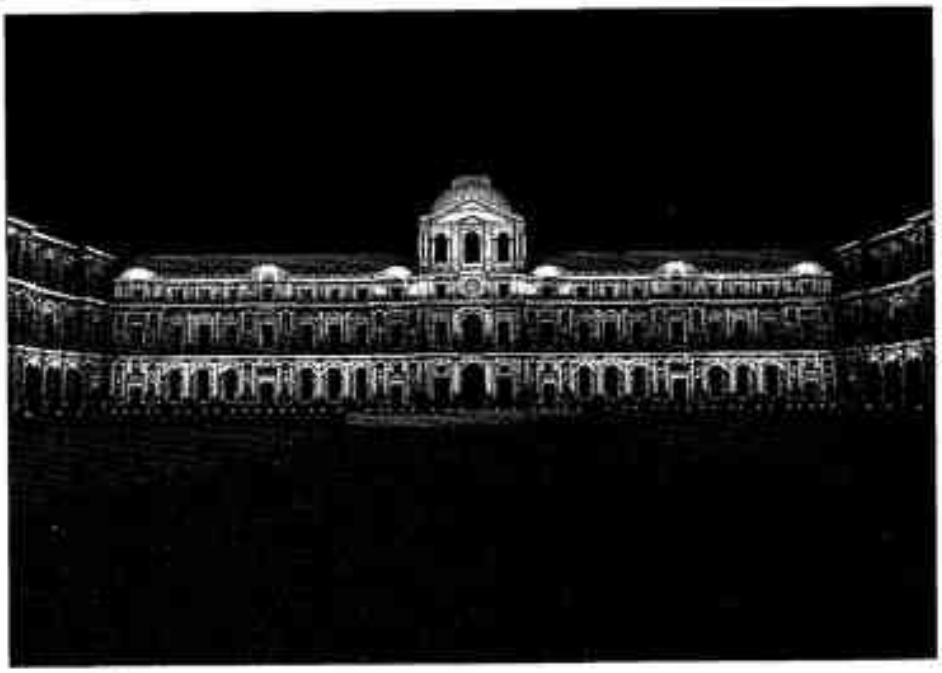
La synthèse d'images a connu, au cours des dernières années, un développement remarquable, tant pour ce qui est des fonctionnalités qu'elle offre à ses utilisateurs, que pour la variété des domaines d'utilisation. Alors que le moteur principal des développements était clairement il y a quelques années l'audiovisuel, on assiste aujourd'hui à une diversification sensible des besoins et des demandes, sans doute parce que le niveau de performance qu'il est possible d'atteindre permet de réaliser, ou d'envisager pour un proche avenir, des applications nouvelles nombreuses. Une telle évolution s'est effectuée sous l'effet conjugué de la baisse des coûts des matériels graphiques, de l'augmentation considérable de la puissance disponible (tant en rapidité de calcul, qu'en capacité et qu'en efficacité graphique) et du développement de systèmes logiciels aux possibilités de plus en plus riches. Elle a été soutenue par une activité de recherche extrêmement vivante, œuvre de laboratoires industriels comme de laboratoires publics.

Complexité, efficacité, interactivité

On assiste aujourd'hui à une course vers trois objectifs principaux qui sont à la fois complémentaires et, dans une certaine mesure, contradictoires. Ce sont : la recherche de la maîtrise de modèles de plus en plus complexes, la recherche d'une efficacité de plus en plus grande et enfin l'ouverture des systèmes vers l'utilisateur par une interactivité de plus en plus importante, permettant d'intervenir très rapidement au cours du déroulement du système. Nous allons développer brièvement ce que recouvrent ces trois notions, afin de donner un aperçu très bref de certaines directions suivies par la synthèse d'images.

La maîtrise de la complexité se traduit par le fait qu'on est capable de représenter aujourd'hui de manière statique, mais aussi d'animer, des scènes comportant un nombre impressionnant de composants géométriques primitifs (plusieurs centaines de milliers, voire plusieurs millions de polygones, par exemple). Il faut toutefois noter que, à moins de disposer de techniques procédurales qui construisent elles-mêmes les objets complexes (souvent en utilisant la récursivité nichée à "un peu d'aléatoire"), le travail de construction de telles scènes est terriblement long et requiert le plus souvent des mois de travail, par exemple pour des représentations architecturales fidèles. Des techniques plus automatiques (reconstructions à partir de prises de vue, etc.), développées dans certains laboratoires, devraient permettre des progrès sensibles dans le futur.

La maîtrise de la complexité se traduit aussi par le fait que l'on modélise des phénomènes de plus en plus complexes, d'une façon de plus en plus fine. Les *modèles de la lumière* en sont un bon exemple : très primitifs il y a une dizaine d'années (ce qui donnait à tous les objets



Simulation numérique et visualisation du projet d'illumination de la Cour carrée du Louvre (simulation réalisée dans le cadre d'une opération de mécénat technologique et scientifique d'EDF en partenariat, dans le cadre de l'association ADMITECH, avec le Centre de recherche en informatique de Nancy et le CNAI, École d'architecture de Nancy).

soit l'apparence du plastique, soit des reflets très métalliques), ils sont devenus aujourd'hui d'une extrême précision, tant au niveau local (propriétés locales des matériaux vis-à-vis de la lumière qu'ils reçoivent et renvoient) que global (prise en compte des effets des objets les uns sur les autres, inter-réflexions, etc.). Au point que, parfois, il est difficile de reconnaître une image de synthèse d'une photographie. Des simulations aussi réalistes ne sont pas seulement intéressantes pour les images qu'elles permettent de produire, mais aussi, et surtout, pour les quantités physiques qu'elles permettent de calculer, tirant ainsi le domaine vers des applications techniques qui dépassent la simple image. Un autre excellent exemple des progrès accomplis en matière de modélisation d'objets complexes est celui de la modélisation des plantes et des arbres. Les images que l'on sait produire sont, là aussi, très impressionnantes de réalisme (dans un grand nombre de cas, la quête de réalisme et la quête de complexité se rejoignent). Mais les plus intéressantes sont celles qui reposent sur des *modèles botaniques* qui, au-delà de l'image, et en prenant en compte des paramètres qu'il est possible d'aller mesurer (de manière statistique) sur le terrain, permettent de prévoir des évolutions, de calculer des productions agricoles, etc. De nombreux autres phénomènes naturels (feux, chutes d'eau, brouillard, nuages, etc.) ont donné aussi lieu à des travaux de représentation.

La simulation du mouvement

Un domaine différent, celui de la simulation du mouvement, est l'objet de recherches très actives. Les systèmes d'animation traditionnels reposent sur l'interpolation entre des images-clés. Pour reproduire des mouvements complexes, il est donc nécessaire de produire à la main, c'est-à-dire par placement interactif, un nombre énorme de positions-clés. On assiste depuis peu au développement de techniques qui sont qualifiées de *reposant sur la physique*, parce qu'elles utilisent les notions de masse, force, couple, etc., et, pour certaines d'entre elles, des approches traditionnelles en mécanique du solide ou en mécanique des milieux déformables. De telles approches sont extrêmement prometteuses pour simuler des mouvements non triviaux d'objets complexes déformables (déformations, chocs, réponses aux chocs, etc.). Il faut souligner que ce que l'on recherche le plus souvent, et ceci n'est pas valable seulement pour l'animation, est un réalisme visuel. La précision que l'on attend des *modèles physiques* n'est donc, en général, pas la même que celle recherchée dans leur domaine d'origine. De plus, l'exigence d'interactivité, presque toujours présente, conduit à simplifier ou modifier les modèles ou même, très souvent, à en adopter d'autres, mieux adaptés aux buts recherchés. L'étude que l'on fait des phénomènes est, en effet, bien souvent très différente de celle faite dans les disciplines classiques qui les étudient. La prise en compte de contraintes dans les mouvements à calculer (éviter des obstacles, faire suivre une trajectoire prédéfinie à un point, etc.) est aussi un problème important et difficile auquel on ne sait apporter que des réponses partielles. Le résoudre fournira non seulement des outils permettant de construire des animations très difficiles à réaliser – voire impossibles dans certains cas – par les moyens traditionnels, mais aussi des techniques applicables en robotique par exemple, dans un domaine encore tout neuf qui est celui de la robotique des objets flexibles.

Des calculs massivement parallèles

La recherche de l'efficacité a été un souci permanent, et bien naturel, pour ceux qui ont développé des systèmes graphiques. Après avoir conduit il y a quelques années au développement d'architectures spécialisées, elle conduit plutôt aujourd'hui à s'intéresser à des *modèles de calcul* différents (parallélisme massif, calcul distribué, etc.), s'interroger sur leur apport, repenser les problèmes – ou plutôt les solutions – en fonction de ces architectures. Comme dans d'autres domaines, il apparaît que le calcul massivement parallèle, par exemple, se prête très bien à des simulations relativement naïves, reposant sur des discrétisations massives des objets représentés (systèmes masses-ressort, par exemple, pour certains matériaux déformables que l'on veut "animer" comme l'on dit en synthèse d'images). Il est important de remarquer aussi que l'efficacité de la seule phase de simulation n'a souvent que peu d'intérêt. Comme il a été dit plus haut, la création, la modélisation d'objets complexes requièrent encore aujourd'hui un temps considérable et ce qu'il est intéressant de chercher à réduire, c'est le temps total de la création suivie de la simulation proprement dite. De ce point de vue, le calcul massivement parallèle offre la possibilité de mettre directement en œuvre des modèles très naturels.

L'interactivité est désirable pour la richesse des possibilités qu'elle offre, qui est évidente, et aussi pour l'économie qu'elle peut procurer. Elle permet, par exemple, d'intervenir rapidement pour arrêter la simulation d'un phénomène dont les premières phases de la simulation montrent clairement qu'elle ne correspond pas à ce qui est souhaité. Ici encore, on peut considérer que rajouter de l'interactivité (ce qui est souvent loin d'être trivial, l'introduction d'événements extérieurs aux simulations posant parfois des problèmes difficiles) permet d'atteindre une plus grande efficacité sur le temps total de création et de simulation. Qui dit interactivité dit aussi nécessairement interface homme-machine, et il est clair que la façon dont est réalisée cette interface influe de manière essentielle sur la facilité d'utilisation et, par là, l'intérêt du système. Il est à prévoir que l'utilisation de nouveaux intermédiaires entre l'homme et la machine, tels que ceux (gants, écrans spéciaux, systèmes capables de réagir, etc.) dont on parle aujourd'hui quand on prononce le terme *réalité virtuelle*, aura une forte influence sur le domaine par les possibilités beaucoup plus grandes qu'ils donneront d'agir directement sur les objets.

Les progrès de la synthèse d'images reposent en partie sur la prise en compte des évolutions et progrès de plusieurs domaines de l'informatique. Mais la spécificité du point de vue qui vise à proposer *in fine* une perception visuelle des phénomènes, est de chercher en permanence à obtenir une plus grande complexité des modèles, une plus grande efficacité des simulations et une plus grande interactivité des systèmes. Cela conduit à privilégier certaines approches, développer certaines techniques spécifiques, accepter des compromis en gardant en tête que c'est une combinaison des trois objectifs qu'il convient d'optimiser.

La synthèse d'images est arrivée à un stade où des outils réellement puissants sont en cours de développement et peuvent être mis au service de disciplines qui n'y avaient pas nécessairement recours jusqu'ici. Elle devrait devenir un intermédiaire essentiel permettant aux scientifiques, chercheurs, ingénieurs, d'appréhender des situations, des phénomènes complexes et évolutifs, sans parler d'applications plus traditionnelles telles que l'audiovisuel, la création artistique, etc. L'efficacité de l'interaction entre l'utilisateur et le système jouera un rôle de plus en plus grand et permettra aux spécialistes de participer de manière active à certaines étapes de la création ou de la simulation des phénomènes qu'ils cherchent à produire ou reproduire. Les systèmes en seront plus efficaces, mais aussi sensiblement plus intéressants à utiliser de par la richesse des situations qu'ils permettront de créer.

Claude Puech, professeur à l'Université Joseph Fourier, équipe IMAGIS, IMAG, 46, avenue Félix-Viallet, 38000 Grenoble.

LA COMMUNICATION GRAPHIQUE

L'apparente simplicité avec laquelle on peut produire des images par ordinateur conduit à vouloir en faire un grand usage : tout le monde connaît l'adage "une image vaut mieux que mille mots", et veut l'illustrer... Cependant, un minimum de réflexion et de formation s'imposent. En effet, ce n'est pas parce qu'une image est produite par ordinateur qu'elle a qualité à être une référence absolue (l'ordinateur a dit...). Encore faut-il identifier le rôle qu'on veut lui faire jouer.

On distingue généralement trois types d'images :

- *Les images de construction*, qui permettent de définir les objets que l'on souhaite traiter : c'est un échange de l'homme vers la machine.

Ces images peuvent correspondre à des procédés interactifs, comme dans le cas de la Conception Assistée par Ordinateur. On cherche alors à faciliter le travail de l'opérateur, quitte à lui masquer (par exemple) le modèle géométrique réellement utilisé. Si l'on utilise des procédés automatiques (cas du génie médical), la difficulté sera d'identifier correctement les composants à traiter. Dans tous les cas, l'image de construction doit permettre de définir sans ambiguïté les objets, éventuellement de manière schématique, afin de s'exprimer le plus simplement et le plus rapidement possible. En ce sens, elles doivent rester proches de la culture et des connaissances habituelles de l'utilisateur : un styliste sait juger visuellement les courbes d'une carrosserie de voiture, mais n'est pas forcément sensible aux notions de dérivée seconde et plus.

- *Les images de compréhension*, destinées à permettre la prise de connaissance d'une situation donnée : c'est un échange de la machine vers l'homme. C'est ainsi que tous les graphiques de décision (courbes, histogrammes, camemberts et autres présentations, cartes diverses et variées) sont certainement un instrument de découverte extraordinaire.

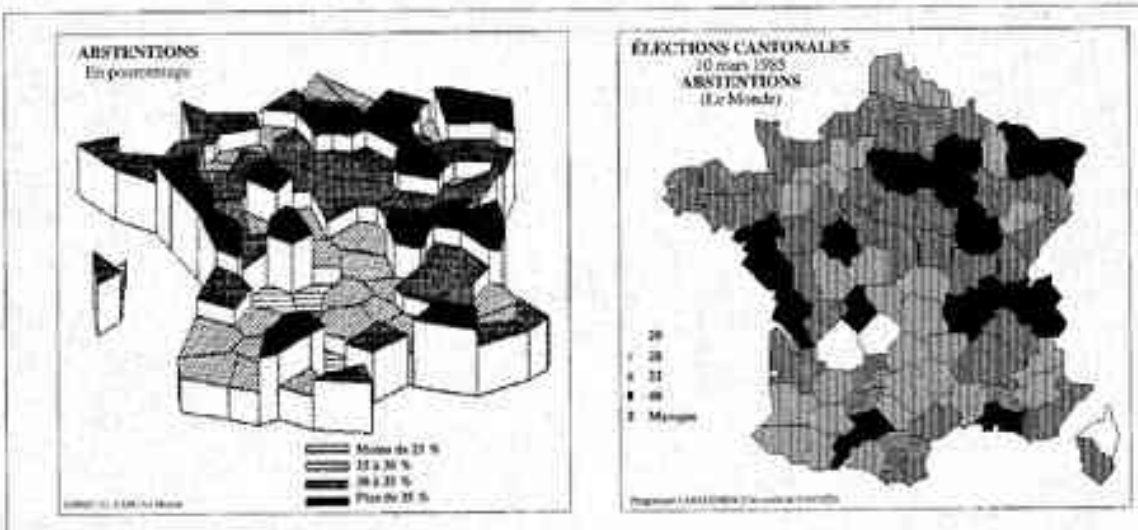
L'apport de l'ordinateur doit résider dans sa rapidité à proposer une présentation graphique correcte, c'est-à-dire ne biaisant pas l'information traduite visuellement. Sur ce point, que d'erreurs sont commises, rendant la lecture des images soit impossible (moindre mal en un sens...), soit dangereuse (si l'on est conduit à une interprétation visuelle fausse par rapport à la vérité des données). De même, l'utilisation des derniers cris de la technique pour présenter n'importe quoi ne paraît pas indispensable : quel sens peut-on donner à la présentation "réaliste" (scènes tridimensionnelles avec ombres portées, transparences, reflets et autres gadgets) d'objets typiquement bidimensionnels comme des résultats électoraux (voir figure) ?

- *Les images de communication*, qui correspondent à des échanges entre personnes (peut-être par l'intermédiaire d'un ordinateur). Elles sont destinées à mettre en valeur une idée, un résumé de données, à frapper l'imagination. Tout est alors permis (ou presque). C'est ainsi que l'usage de la couleur, tellement difficile à utiliser pour la compréhension, peut ici trouver sa véritable dimension. La simplification de l'image pour mettre en valeur ce qui est essentiel est, par exemple, une technique de communication tout à fait valide, mais elle devrait être interdite dans une phase de compréhension !

Autant on peut être un partisan convaincu de l'utilisation raisonnée de l'image, autant il faut être un farouche adversaire de son utilisation passe-partout. Le livre de Bertin "la graphique et le traitement graphique de l'information" (Flammarion, 1974) est édifiant à ce sujet. Il est à la base de nombreuses expériences de traitement graphique de données (pour dépouiller des questionnaires, par exemple), expériences tentées tant dans l'enseignement primaire que secondaire (et au-delà). Les traitements proposés sont facilement programmables, leur efficacité n'est plus à démontrer. Bien sûr, ce n'est pas aussi spectaculaire que des vues d'objets pseudo-tridimensionnels, mais au moins c'est lisible, efficace, en un mot : utile.

Ainsi donc, l'utilisation de l'infographie devrait passer d'abord par un apprentissage : comment bien former une image, comment lire celle-ci, savoir déjouer les pièges visuels, rester critique devant une présentation graphique "léchée". Ce n'est qu'à ce prix que l'image sera instructive, et non simplement décorative.

Michel Lucas, professeur à l'École Centrale de Nantes, Institut de recherche en informatique de Nantes, 1, rue de la Noë, 44072 Nantes Cedex 03.



La carte de droite a été établie à partir des données relevées sur la présentation de gauche. Le manque de certaines données (représenté par « sur la carte) provient de la disparition de certains départements sur la présentation. Faut-il incriminer les résultats électoraux des départements voisins ?

LE GESTE ET LE TOUCHER

La communication gestuelle est la grande oubliée de la communication homme-machine. Des raisons culturelles, mais aussi techniques, expliquent son absence. Les développements actuels devraient lui permettre de trouver la place qui lui revient à côté de l'ouïe et de la vue.

Claude Cadoz

BODY MOTION AND TOUCHING - It will soon be possible to associate the body motion/touching combination with the other sensorial channels, hearing and sight. Communication with machines will become multisensorial. The development of a force feedback gesture transducer is a fundamental prerequisite for this to be possible.

Nous sommes depuis longtemps familiarisés avec la téléphonie et la télévision, mais l'idée d'une télé-transmission de phénomènes concernant d'autres sens que l'ouïe et la vue ou encore de nos actions physiques, nos gestes, est beaucoup moins naturelle. La télé-manipulation existe pourtant depuis les années 60, quand des expérimentateurs de laboratoires nucléaires utilisaient des systèmes articulés très précis, permettant à un bras mécanique prolongé d'une pince de reproduire les mouvements de l'opérateur.

A l'inverse de ses aînés, le canal gestuel n'a pas trouvé son essor et la poignée de main à distance n'est pas près de se substituer au traditionnel "Allo !".

Le franchissement d'une autre distance, celle qui sépare le monde humain naturel des mondes artificiels de la machine et de l'informatique ou bien de certains mondes inaccessibles ou hostiles comme les mondes sous-marins, les environnements radio-actifs... interviendra d'abord.

Manipuler les molécules avec une pince

Imaginez un univers submicroscopique de molécules organiques complexes. Par le moyen de synthèses d'images tridimensionnelles et en exploitant les propriétés connues des atomes et de leurs interactions, ce monde peut être représenté sur un grand écran couleur. Saisissez la poignée qui se balance devant vous, articulée dans toutes les directions et attachée à l'extrémité d'un bras mécanique muni d'un gros moteur à chacune de ses articulations. Vous apercevez une petite molécule qui arrive vers une plus grosse. Cette petite molécule vous la tenez dans la main. Lorsque vous la placez dans la bonne position et qu'une liaison chimique est possible, vous sentez les forces de répulsion s'atténuer et vous trouvez "au toucher" la bonne liaison chimique.

Vous êtes au Département d'informatique de l'Université de Caroline du Nord, à Chapel Hill. Le dispositif, appelé système Grope, que vous utilisez est une application de la communication par le geste et le toucher à ... la chimie moléculaire.

En France, le groupe de l'ACROE, au Laboratoire d'informatique fondamentale et d'intelligence artificielle de l'IMAG, travaille depuis de nombreuses années sur des dispositifs spéciaux, des interfaces bidirectionnelles pour le canal gestuel ou transducteurs gestuels rétroactifs. Grâce à ces dispositifs, on manipule directement des objets comme des cordes vibrantes que l'on peut pincer, frotter, percuter, des marionnettes que l'on peut animer, etc. Mais surtout, grâce à des moteurs très spéciaux intégrés dans la structure de ces interfaces, il est possible de sentir dans ses doigts tous ces objets comme s'ils étaient physiquement présents alors que ni les uns ni les autres ne sont réels, mais tout simplement... simulés, synthétisés par des programmes, des calculs numériques dans un ordinateur.

En robotique, on parle de télé-présence. A l'*Advanced robotic research Ltd*, près de Manchester, pour ne citer que ce lieu, on étudie des robots qui peuvent être envoyés pour des missions spéciales dans des milieux hostiles à l'homme : fonds marins, espace, centrales nucléaires. Tandis que tout ce que ces robots peuvent "voir", "entendre", "manipuler" est transmis grâce à des caméras, microphones, capteurs de forces, etc., à une station de contrôle, l'opérateur revêt diverses prothèses, gants de données et autres exo-squelettes qui captent ses mouvements propres. Il peut alors quasiment s'identifier au robot et s'imaginer être présent dans son environnement, pour y opérer effectivement.

Les réalités virtuelles

Depuis quelques temps, le terme médiatique de réalité virtuelle venu des Etats-Unis fait fureur. La réalité en question est celle d'un environnement artifi-

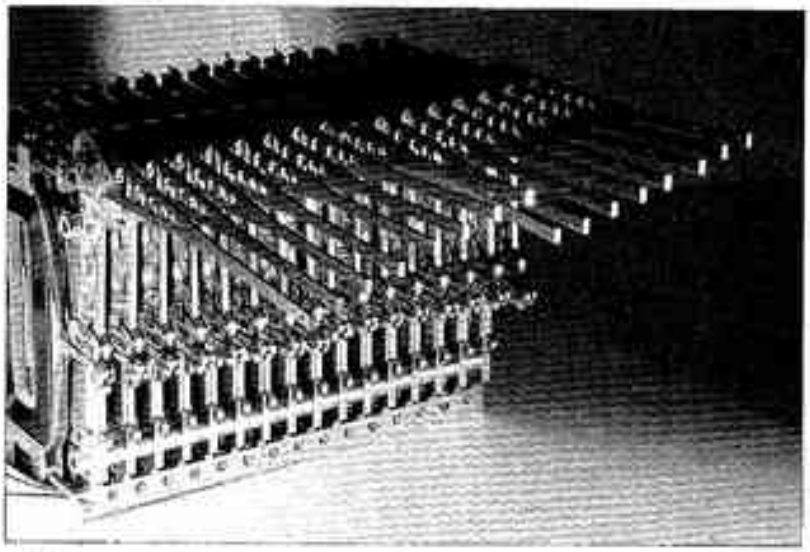


Fig. 1 - Transducteur gestuel rétroactif de l'ACROE. Clavier à 16 touches dotées d'un moteur pour la production du retour tactile : vue du système sans son habillage (© ACROE / LIFIA).

ciel produit par synthèse d'images tridimensionnelles, présentées en stéréovision à un observateur par une paire de télévisions miniatures qu'il chausse comme des lunettes. L'observateur "entre" dans cet espace et s'y "déplace". Ce n'est ni une magie ni une métamorphose, c'est une illusion dont la force tient à un principe simple : l'opérateur porte un autre dispositif qui capte les mouvements de sa tête et les envoie à l'ordinateur ; celui-ci recalcule les images à chaque instant de telle façon qu'il y ait concordance entre leurs modifications et celles qui se produiraient s'il s'agissait d'un espace réel. Dans le même temps, par sa main gantée d'un gant numérique qui capte le mouvement de ses doigts, le même sujet peut agir, par des transpositions de ses mouvements manuels, sur des éléments de cet environnement. Il faut noter que, sauf pour quelques dispositifs très récents et de performances encore très limitées, l'opérateur n'a pas, dans ces réalisations, de sensation tactile dans le gant numérique et qu'il ne reçoit en conséquence aucune information de la part des objets. La communication gestuelle reste ici unilatérale.

Tous ces exemples, même s'ils révèlent qu'il y a encore du chemin à parcourir, attestent de l'entrée du geste et du toucher dans le monde de la communication. Mais pourquoi cette entrée est-elle tardive et sa destinée différente ?

Il y a tout d'abord une raison culturelle, comme le souligne Y. Hatwell du Laboratoire de psychologie expérimentale à l'Université de sciences sociales de Grenoble, qui fait remarquer que dans les civilisations occidentales, le contact physique est socialement réprimé. Le canal gestuel n'est pas considéré comme un canal naturel de communication.

Le geste est pourtant, à côté de la voix, un moyen fondamental d'informer son environnement. Il est par ailleurs le seul et unique moyen de le transformer matériellement. Le geste peut revêtir des fonctions extrêmement différentes, depuis celle de langage (le langage des sourds-muets, ou celui du chef d'orchestre) jusqu'à celle où, pressant les touches d'un clavier, il transmet de l'information, sans oublier bien sûr les fonctions essentielles de déplacement de charge, de commande d'un véhicule ou de mise en vibration des cordes d'un violon.

Toucher mais aussi sentir

L'action est, en pratique, indissociable de la perception tactilo-kinesthésique. L'absence de cette dernière est responsable de la frustration que l'on ressent dans la communication avec l'ordinateur. Le "clavier fait écran". Dans la relation naturelle avec l'environnement, le toucher joue un rôle complémentaire des autres sens. La vision opère en un champ continu où le soin d'isoler perceptivement un objet revient à des processus d'ordre supérieur. La perception tactilo-kinesthésique permet d'isoler l'objet soit dans le monde physique (en le saisissant), soit à un niveau perceptif tout à fait périphérique (en posant la main ou les doigts là où c'est utile). L'exploration tactile est discontinue et limitée aux objets proches, mais une fois cette sélection opérée, par le fait qu'il y a contact matériel, la communication est la plus intime qui se puisse. Les sons, à l'opposé des images et à l'action des impressions tactiles, nous parviennent sans que nous ayons à effectuer la moindre action de visée ou d'exploration. La complémentarité des sens va très loin : on a pu vérifier, par des expérimentations appropriées, que l'on ne voyait pas nécessairement la même chose si l'on associait à la vision certaines perceptions acoustiques plutôt que d'autres, certaines perceptions tactiles plutôt que d'autres, que l'on n'entendait pas la même chose si l'on associait à l'audition certaines images plutôt que d'autres. Dans ce jeu inter-sensoriel, la perception tactilo-kinesthésique a un rôle non négligeable.

Mais ce n'est pas seulement à cause du tabou que le geste et la perception tactilo-kinesthésique ont été relégués au second plan, c'est aussi parce que les phénomènes gestuels, plus "matériels" que le son et l'image, sont aussi plus difficiles à capter, convertir en information et restituer.

Transformer nos gestes en signaux électriques

Pour le canal gestuel, nous devons disposer de transducteurs appropriés, capables de transformer nos gestes en signaux électriques qui puissent être assimilés par les ordinateurs ou les machines de transmission, et de transformer des signaux électriques en phénomènes tactilo-kinesthésiques. Cette seconde fonction ne peut être réalisée qu'avec des moteurs électriques aux performances très particulières, à la fois assez puissants et assez rapides, installés sur des mécanismes articulés complexes. Et cela n'est pas véritablement aisé.

Le développement des transducteurs gestuels à retour d'effort (voir figures), comme on doit appeler ces dispositifs, est la condition d'établissement de la communication par le geste et le toucher. Les perspectives sont aujourd'hui très encourageantes et l'on peut imaginer que dans un avenir assez proche, une bonne part de la communication avec la machine ou, par son intermédiaire, avec l'environnement physique ou humain, pourra se dérouler selon un mode véritablement multisensoriel où la vue, l'ouïe et le toucher seront réconciliés et réunifiés. La combinaison du geste et du toucher avec les autres canaux sensoriels ouvre la voie à un monde de dialogue où, à côté de signes et de symboles, pourraient se trouver dans un environnement familier des objets très concrets avec non seulement une forme visuelle comme les icônes du Macintosh, mais aussi un poids, une matière, un timbre ...

Claude Cadoz, ingénieur de recherche, directeur de l'ACROE. Laboratoire d'informatique fondamentale et d'intelligence artificielle (URA 394 CNRS), 46, avenue Félix Viallet, 38000 Grenoble.

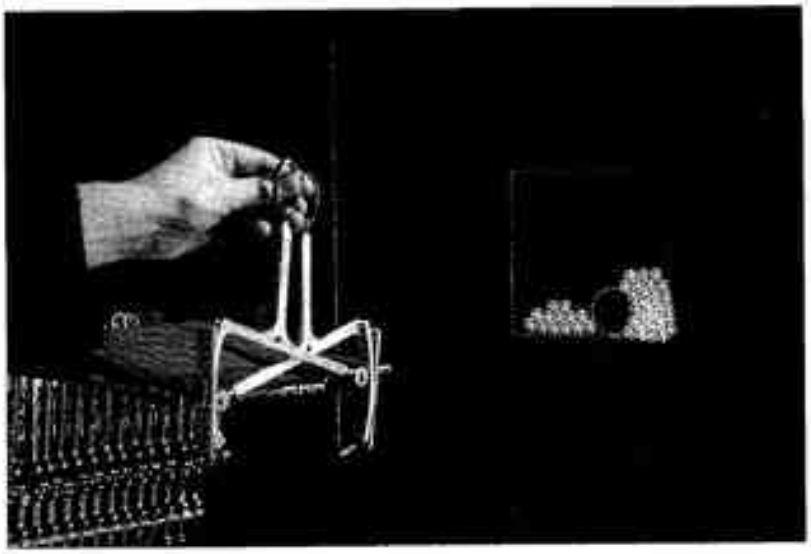


Fig. 2 - Système de la figure 1 monté en "pièce à quatre degrés de liberté" (© ACROE / LIFIA)

PRODUCTION DE DOCUMENTS ET PAO

La publication assistée par ordinateur se contente encore trop souvent de simuler les opérations faites classiquement sur la table de montage. Il existe pourtant d'autres possibilités.

Vincent Quint

PRODUCING DOCUMENTS AND DTP - The advanced versions of DTP (desktop publishing) take into account the formal structure of documents. In the future, sound and moving images could also be integrated. As a result, DTP would become multimedia.

Les premiers outils informatiques pour la production de documents réalisaient une simulation des outils classiques : les traitements de texte simulaient une machine à écrire, les systèmes de PAO (publication assistée par ordinateur) simulaient une table de montage. Les outils actuels sont marqués par cette origine et offrent encore peu de fonctions réellement nouvelles. Leur intérêt vient plutôt du support électronique qui apporte rapidité et souplesse, mais ils profitent peu des possibilités de *traitement* de l'ordinateur. La plupart des recherches menées dans le domaine visent à combler cette lacune.

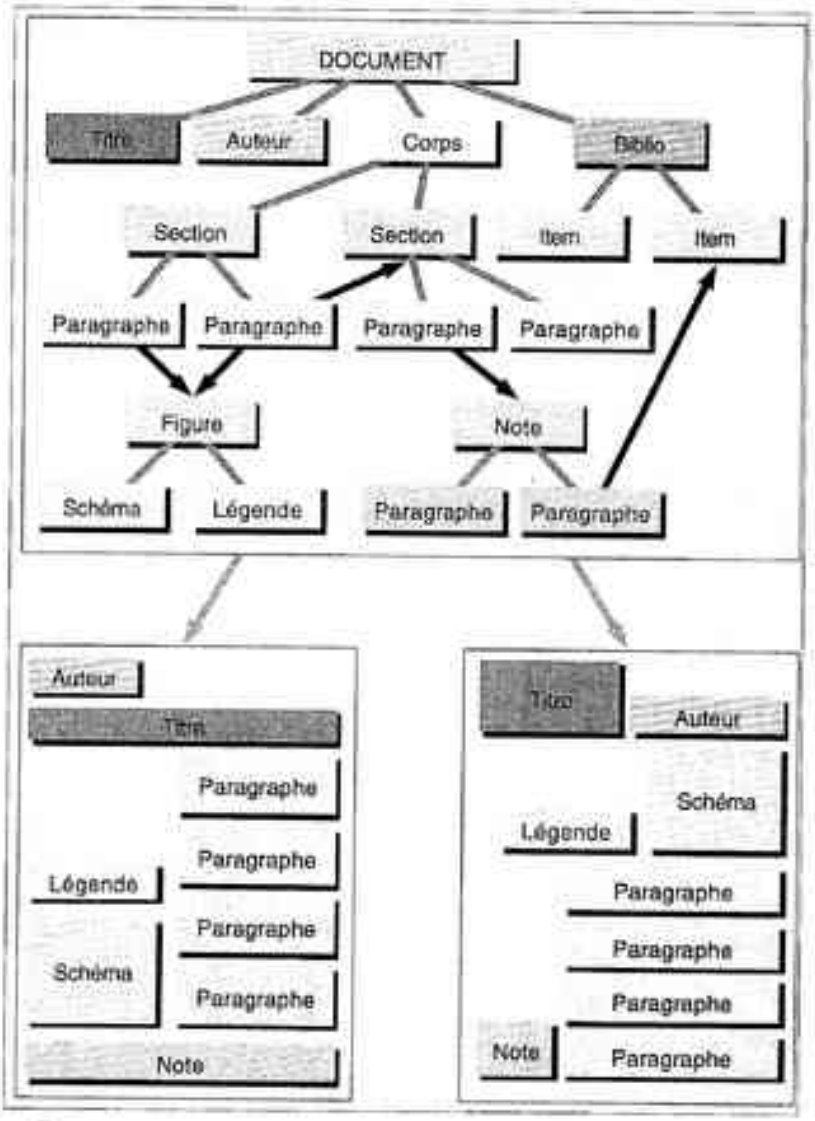
L'étude de nouveaux modèles de documents contribue au développement d'outils plus puissants. Le modèle le plus courant aujourd'hui considère d'abord la forme *concrète* (graphique) des documents : un document est une suite de paragraphes qui ont chacun un style (police, interligne, marges, etc.). Au contraire, les travaux actuels s'orientent vers une représentation *abstraite* des documents, où des composants comme les titres, les sections, les notes, les paragraphes, les renvois, etc. sont représentés en tant que tels, ainsi que la structure qu'ils forment. De plus, les différents types de composants disponibles et les relations structurales qui peuvent les relier (ordre, inclusion, référence, etc.) sont déclarés explicitement, selon le type de document à produire.

La structure abstraite des documents

Ainsi, la structure des documents à traiter est formellement définie, ce qui permet de nombreux traitements : à partir de la structure abstraite d'un document, on peut produire automatiquement son aspect graphique, et même plusieurs aspects graphiques différents, on peut élaborer des tables des matières ou des index, on peut maintenir les numérotations et les références croisées, on peut dériver différentes formes de document, on peut alimenter des bases de données, etc. L'un des axes de recherche actuels consiste à étudier de tels modèles et à mieux comprendre les avantages qu'on peut en tirer.

Cette nouvelle conception des modèles de documents a entraîné une évolution importante dans les travaux sur la reconnaissance des documents. Il ne s'agit plus seulement de reconnaître des caractères manuscrits ou imprimés, mais il faut aussi retrouver la structure abstraite des documents analysés pour pouvoir les traiter ensuite efficacement.

Un autre axe vise à formaliser et à intégrer dans les outils informatiques le savoir et le savoir-faire des professionnels qui interviennent dans la chaîne traditionnelle de production des documents. En effet, les outils actuels permettent à une seule personne de jouer à la fois le rôle de l'auteur, du concepteur graphique, du typographe, du correcteur, de l'éditeur et même de l'imprimeur ou du dessinateur de caractères. Mais bien peu d'utili-



A partir d'une représentation abstraite des documents (en haut), de nombreux traitements sont possibles. On voit ici deux mises en page différentes qui peuvent être obtenues automatiquement.

sateurs réunissent toutes ces compétences et les systèmes actuels offrent très peu d'aide dans ces domaines. Tout au plus peut-on citer les correcteurs orthographiques qui visent à pallier l'absence de correcteur (humain), mais dans le domaine de la conception graphique ou de la typographie, beaucoup reste à faire.

Vers une coopération entre logiciels

La plupart des systèmes de production de documents sont des logiciels très fermés : ils permettent à un utilisateur de produire des documents qui sont soit imprimés, soit stockés dans des fichiers. Mais il existe par ailleurs de nombreuses applications informatiques qui utilisent des documents : un atelier logiciel ou un système de CAO produit des programmes ou des plans, mais aussi toute la documentation qui les accompagne et cette documentation, pour être cohérente, doit être produite dans le même environnement. Pour cela, il faut que l'outil de production de documents soit intégré, et d'une façon plus fine que par la simple communication de fichiers. Il faut donc trouver des architectures logicielles qui permettent aux outils de production de documents de coopérer étroitement avec d'autres logiciels.

Il faut enfin citer un axe de recherche prometteur qui vise à considérer les documents d'une façon très large, en intégrant non seulement le texte et le graphique, mais aussi le son et l'image animée ainsi que les relations (temporelles notamment) entre ces supports : c'est l'hyper-média.

Vincent Quint, directeur de recherche à l'INRIA, Bull-Imag/Systèmes (UMR 122 CNRS), 2, avenue de Vignate, 38610 Gières.

LES INTERFACES LOGICIELLES

Couches logicielles entre l'homme et la machine, les interfaces prennent aujourd'hui une importance croissante. Leur construction nécessite des outils modulaires et une méthodologie itérative. Leur extension à plusieurs médias et modalités sensorielles pose des problèmes nouveaux.

SOFTWARE INTERFACES - Software interfaces allow for man-machine communication. They are now modular and produced using tool boxes, skeletal applications, and automatically by interface generators. The development of multimedia systems that combine several sensorial channels raises new, sensitive, fundamental problems.

Joëlle Coutaz

Au sens large, une interface est un dispositif servant de limite commune à plusieurs entités communicantes. Dans le domaine de la communication homme-machine, une *interface logicielle* (ou interface homme-machine) est un ensemble de composants logiciels permettant à un ou plusieurs individus de communiquer avec un système informatique, en vue d'accomplir une tâche. Un tel système, capable de communiquer en temps réel avec des utilisateurs, est dit *interactif*.

L'organisation des composants d'une interface homme-machine (IHM) obéit à des règles prescriptives. Des outils, véhicules de ces éléments directeurs, sont nécessaires au concepteur et au réalisateur de logiciels interactifs. Parallèlement à la mise en place de techniques pratiques, la définition de concepts adaptés à l'évolution du domaine se poursuit dans les laboratoires de recherche.

Les éléments directeurs

La conception logicielle des systèmes interactifs et leur réalisation se heurtent à deux sources de difficultés. D'une part l'environnement inclut des applications aux besoins les plus divers, une grande variété de matériels et des utilisateurs qui ne cessent d'évoluer. D'autre part, le caractère immédiat et informatif des échanges, dans un système interactif, met à contribution tous les composants logiciels, avec pour conséquence le risque de constituer un amalgame confus entre l'expression du comportement perceptible du système et la production des informations qui relèvent du domaine. Cet amalgame rend difficile, voire impossible, l'ajustement du système aux variations de l'environnement.

Jusqu'à ces dernières années, on ne disposait pas de modèle de référence pour la construction des systèmes interactifs. Sans cadre directeur pour appliquer la modularité, les réalisateurs ont créé des logiciels mal adaptés à l'évolution itérative des interfaces.

Les modèles d'architecture proposés actuellement pour les systèmes interactifs reposent tous sur la distinction entre les fonctions spécifiques à un domaine (les services) et les mécanismes de présentation (l'IHM). Nous appellerons désormais :
- *noyau fonctionnel* l'ensemble des services logiciels spécifiques à un domaine, par exemple une base de données mémorisant les horaires d'avion ;
- *interface homme-machine* le logiciel chargé de la communication d'informations entre le noyau fonctionnel et l'utilisateur, par exemple la présentation des horaires sous forme de formulaire ;
- *système interactif* l'ensemble constitué d'un noyau fonctionnel relié à une interface, par exemple un système de consultation d'horaires.

La distinction *noyau fonctionnel, interface homme-machine* sert de fondement à l'ajustement itératif du système : en théorie, la modification de la partie IHM peut être effectuée sans mettre en cause les services du noyau fonctionnel et réciproquement.

Bien que valide en toute circonstance, ce schéma définit une organisation trop générale pour être directement exploitable par le concepteur de logiciel interactif. Des modèles multi-agents affinent cette image en unités à la fois mieux adaptées aux techniques du génie logiciel, mais aussi plus sensibles aux impératifs cognitifs de la communication homme-machine.

Les outils

Les outils pour la construction d'IHM se répartissent en trois catégories : les boîtes à outils, les squelettes d'application et les générateurs d'interfaces.

Une *boîte à outils* est une bibliothèque de procédures. L'éventail des fonctions offertes inclut la gestion des fenêtres, des événements tels que les clics souris ou la frappe de caractères sur le clavier, la production de figures géométriques simples comme les polygones et les cercles, la gestion des entités de dialogue tels les menus, les boutons poussoirs et les formulaires. Parmi les avantages d'une boîte à outils, nous retenons la définition d'un style d'interaction "normalisé" que l'utilisateur retrouve de système en système. Parmi les inconvénients, nous notons

l'absence de guide pour la conception logicielle et un apprentissage difficile.

Un *squelette d'application* réalise les fonctions usuelles de l'interface homme-machine sous la forme d'un logiciel réutilisable et extensible. La tâche du programmeur consiste à greffer sur le squelette les composants spécifiques à son système. Un squelette est construit au-dessus d'une boîte à outils dont il assemble les composants utiles à l'IHM d'une large classe de noyaux fonctionnels. Il a l'avantage de fournir au programmeur une architecture prête à l'emploi, mais il ne masque pas la boîte à outils sous-jacente.

Un *générateur d'interfaces* crée une large part de l'IHM d'un système interactif à partir d'une spécification. Cette interface est ensuite reliée à un noyau d'exécution qui s'apparente à un squelette d'application. La spécification s'exprime dans un langage spécialisé, ou bien s'effectue par manipulation directe au moyen d'un éditeur interactif. L'objectif premier du générateur est le prototypage rapide : permettre la création d'IHM à moindre coût, puis effectuer des tests et ajuster les choix. Certains générateurs sont utilisables par des non informaticiens : le concepteur dessine son interface et le générateur produit le logiciel automatiquement. En réalité, les interfaces ainsi créées sont limitées par la base d'objets de présentation intégrée au générateur (typiquement les fenêtres, les menus, les boutons, les icônes et les boîtes de dialogue). Malheureusement, la généralité et l'extensibilité passent inévitablement par la programmation.

La recherche

La recherche actuelle en interfaces logicielles vise, pour l'essentiel, l'*intégration du savoir pluridisciplinaire*. On observe un fort rapprochement entre disciplines disjointes comme la psychologie cognitive, la linguistique et l'informatique. Ce rapprochement s'effectue dans le cadre de projets européens ou nationaux tels le programme Cognisciences du CNRS et le PRC Communication Homme-Machine. L'objectif est d'intégrer aux concepts et techniques de l'informatique les résultats exploitables de l'ergonomie cognitive.

On observe également un rapprochement entre disciplines de l'informatique telles la communication en langue naturelle parlée et écrite (en génération comme en reconnaissance), la vision par ordinateur et la synthèse d'image, la kinesthésie (geste et retour d'effort). La mise en commun de toutes ces modalités de la communication ouvre un champ nouveau dont on ne peut encore cerner les retombées socio-économiques. En particulier, on observe aux USA la percée fulgurante des réalités virtuelles et artificielles rendue possible par l'amélioration des performances du matériel. Ces systèmes reproduisent électroniquement un monde réel ou créent des environnements imaginaires en y intégrant le sujet humain. Ils s'appuient sur une communication multimédia et/ou multimodale qui combinent l'audition, la parole, la vision et le geste. La distinction entre le multimédia et le multimodal mérite d'être clarifiée.

Multimédia et multimodalité

Au sens général, un média tel l'ordinateur, est un dispositif technique servant de véhicule à l'information. En psychologie, la notion de modalité réfère aux catégories sensorielles humaines. Ainsi l'information transmise ou acquise par un média fait-elle appel à une modalité de la communication humaine. Un dispositif sera multimédia s'il sollicite simultanément plusieurs canaux sensoriels. Dans le cas des systèmes informatiques, il convient d'affiner ces définitions.

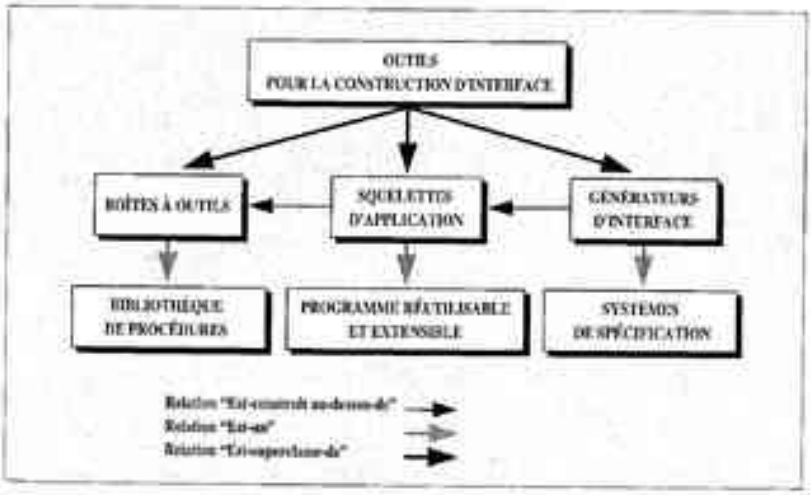
On peut caractériser les systèmes informatiques multimédia et multimodaux selon trois dimensions : pouvoir d'abstraction, fusion et contrainte temporelle.

Le pouvoir d'abstraction traduit la capacité interne du système à découvrir le sens des expressions d'entrée ou à produire des expressions à partir d'une représentation du sens. Le système est donc capable de gérer plusieurs niveaux d'abstraction allant du symbolique au physique. Les systèmes multimédias se limitent aux niveaux d'abstraction les plus bas : bien que les images soient modifiées par des algorithmes de compression, le pixel reste le concept de base. A l'inverse, les systèmes multimodaux visent à découvrir le sens automatiquement pour le mettre en œuvre dans l'interaction homme-machine.

La fusion traduit la combinaison de différents types d'informations. Par exemple, l'incrustation d'images de synthèse dans un film vidéo est une fusion. L'expression "mets ça là" accompagnée des gestes déictiques indiquant l'objet à déplacer et le lieu destinataire, requiert la fusion d'informations d'entrée émises sur des canaux moteurs distincts.

Les contraintes temporelles ont trait à la séquentialité ou au parallélisme dans la production ou la réception des informations. Elles peuvent se situer à différents niveaux de granularité : canal de communication entre unités d'information, au sein d'une unité d'information. Par exemple, le parallélisme au niveau du canal de communication offre la possibilité d'écouter de la musique tout en parlant et en saisissant du texte. Les contraintes temporelles entre unités d'information expriment la synchronisation, par exemple entre des images et du texte. Au sein d'une unité d'information, elles expriment les évolutions entre les entités de cette unité, par exemple les objets constitutifs d'un dessin animé.

On observe dans les applications actuelles que les informations multimédias constituent souvent l'objet de la tâche, alors que les énoncés multimodaux servent à contrôler le déroulement de la tâche. Il est raisonnable de penser que cette distinction s'estompera avec les progrès de la communication multimodale. Toutefois, il faut noter que nous sommes loin de maîtriser ce nouveau type d'interaction, aussi bien sur le plan cognitif que technique. A ce titre, il constitue un champ d'étude prometteur.



Les classes d'outils pour la construction d'interfaces logicielles.

Joëlle Coutaz, professeur à l'Université Joseph Fourier, Laboratoire de génie informatique de Grenoble (URA 398 CNRS), Domaine universitaire de Saint-Martin-d'Hères, Bât. 385, BP 53X, 38041 Grenoble Cedex.

LES ENVIRONNEMENTS INTERACTIFS D'APPRENTISSAGE AVEC ORDINATEUR

Concevoir des systèmes informatiques interactifs pour aider des individus à apprendre, tel est le but général visé en EIAO. Il s'avère nécessaire dans ce domaine de coopérer avec des disciplines aussi variées que la psychologie, la didactique, les sciences de l'éducation et de la communication.

INTERACTIVE LEARNING ENVIRONMENTS - Artificial intelligence and especially knowledge-based techniques are used to improve the capacities of classical computer-aided instruction systems. Intelligent tutoring systems, a special kind of ILEs with a four modules architecture, are based on guided problem solving activities. Multi-disciplinary collaborative projects are necessary to go further.

Monique Baron

L'enseignement intelligemment assisté par ordinateur (EIAO) est né dans la décennie 1970-1980, avec l'objectif de dépasser, en utilisant des techniques d'intelligence artificielle, certaines limites de l'enseignement assisté par ordinateur (EAO) classique. Il visait notamment une plus grande souplesse du déroulement d'une session, davantage d'initiative pour l'élève ainsi qu'une meilleure capacité du système à analyser les réponses de celui-ci. Le développement de ce domaine est fortement lié à celui des "systèmes à base de connaissances" en intelligence artificielle. Les travaux qui s'y rattachent concernent plusieurs types de problèmes, parmi lesquels :

- la représentation explicite de connaissances du domaine enseigné et la modélisation des raisonnements utilisant ces connaissances, pour donner au système la capacité de résoudre de manière communicable des problèmes posés à (ou par) l'apprenant et de fournir des explications à sa demande ;
- la représentation, la mise à jour et la gestion par le système d'informations sur l'apprenant, afin d'assurer une meilleure compréhension de son comportement et une bonne adaptation de l'interaction ;
- la représentation explicite de connaissances pédagogiques, pour assurer la conduite de la session de manière souple, adaptable et efficace ;
- la recherche de modes d'interaction facilitant la communication apprenant-système et favorisant les apprentissages.

Ces problèmes sont diversement posés, selon les objectifs et le type "d'environnement interactif d'apprentissage avec ordinateur" (signification plus récente du sigle EIAO).

Les systèmes tutoriels intelligents

Pour les systèmes tutoriels intelligents, les interactions système-apprenant sont organisées autour d'activités de résolution d'exercices, plus ou moins aidées ou contrôlées par le système qui joue ainsi un rôle de précepteur attentif et plus ou moins dirigiste. L'architecture informatique de

Comment favoriser des apprentissages ? Le dialogue socratique inaugural est une source d'inspiration pour certaines approches en EIAO. Art seldjoukide, 13ème siècle, remise Topkapi, Istanbul ; "Meilleures sentences et plus précieux dictons". Socrate et ses élèves. © Giraudon.

ces systèmes comporte généralement quatre modules interdépendants, qui correspondent respectivement aux quatre types de problèmes mentionnés ci-dessus : module domaine, modèle de l'élève, module pédagogue (ou tuteriel) et interface; ces modules doivent de plus être intégrés dans une architecture adaptée pour coopérer efficacement.

D'autres approches existent pour des EIAO basés sur d'autres activités susceptibles de favoriser des apprentissages : l'exploitation de simulations, l'exploration de micro-mondes, la découverte guidée, l'apprentissage par collaboration par exemple.

La nécessité d'une collaboration pluridisciplinaire

La conception d'EIAO nécessite une large collaboration pluridisciplinaire pour élaborer des modèles et des représentations qui soient adaptés aux domaines de connaissances et aux activités de coopération homme-ordinateur relatifs aux objectifs d'apprentissage visés. Sont ainsi particulièrement concernées, outre plusieurs domaines de recherche en informatique :

- la discipline enseignée elle-même, sa didactique et plus généralement les sciences de l'éducation pour la spécification de situations de formation et de contenus de connaissances ;
- la psychologie cognitive, pour la modélisation d'activités de résolution de problèmes et des processus d'apprentissage humain, ainsi que pour l'étude des situations d'interaction homme-ordinateur ;
- les disciplines liées à la communication homme-ordinateur, telles que la linguistique pour la modélisation de dialogues et le traitement du langage naturel, l'ergonomie cognitive pour l'étude des représentations à l'interface, etc.

C'est dans le cadre de projets pluridisciplinaires organisés autour de prototypes qu'il est possible d'affiner les analyses, les hypothèses et les objectifs concernant les EIAO, et de contribuer ainsi à l'étude et la formalisation des processus cognitifs humains ainsi que des processus de formation. Les travaux en EIAO se rattachent donc à de nombreuses problématiques actuelles en intelligence artificielle et en sciences cognitives.

Monique Baron, maître de conférences à l'Université Pierre et Marie Curie, Laboratoire formes et intelligence artificielle (URA 1095 CNRS), Université Pierre et Marie Curie, Tour 46, 4, place Jussieu, 75252 Paris Cedex 05.

L'ERGONOMIE DE LA COMMUNICATION HOMME-MACHINE

Une science nouvelle, l'ergonomie de la communication homme-machine, est née avec l'ordinateur. Elle tend à instaurer un délicat équilibre entre les exigences de la nature humaine et celles des programmes informatiques.

René Amalberti

THE ERGONOMICS OF MAN-MACHINE COMMUNICATION - There is a very strong demand for ergonomics in man-machine communication. Perhaps too much is expected from it. In modern cognitive ergonomics, ergonomists derive means of communication from actual situations, thereby adopting the opposite approach to that of the engineers who develop increasingly diversified means of communication.

La demande pour une ergonomie de la communication homme-machine (CHM) est considérable. Les systèmes professionnels ou grand public sont d'une complexité croissante et l'on voudrait que leur compréhension par les utilisateurs soit non ambiguë, facile, rapide, si possible sans apprentissage (particulièrement pour les systèmes grand public). L'ergonomie de l'interface est devenue la variable-clé du succès et, souvent encore, la variable sur laquelle on veut et on croit pouvoir agir à l'infini pour remédier aux défauts constatés dans l'exécution de la tâche, sans remettre en cause le cœur technologique du système.

Les systèmes ont eux-mêmes évolué sous la pression de la CHM. Il suffit de regarder le tableau de bord d'un Airbus A320 truffé de moyens de visualisation intégrés, de constater les changements étonnants induits par les commandes électriques de vol, dirigées par des mini-manches latéraux, et d'écouter les alarmes vocales dans le cockpit pour savoir que la CHM dans un avion actuel n'a rien de commun avec celle d'un avion de la génération précédente.

L'évolution des moyens de communication se poursuit de façon spectaculaire. Certaines recherches ou maquettes de laboratoire font rêver : commandes virtuelles (l'exemple du gant numérique est sans doute le plus connu), mondes virtuels de l'image, du son, désignation d'action par poursuite visuelle ou par simples mouvements de tête, dialogue vocal en langage naturel non restreint, etc.

Les tâches et les moyens

L'ergonomie ne crée pas elle-même ces moyens. Elle tente seulement de les organiser et de les coordonner pour atteindre le meilleur résultat. La multiplication et la variété des moyens actuels est à la fois une aide et une contrainte nouvelle : l'ergonomie moderne de la CHM ne doit plus seulement optimiser un moyen de dialogue. Elle vise à l'intégration, souvent très complexe, de *plusieurs* moyens (dialogue multimédia) avec pour but de diminuer l'asymétrie du dialogue entre l'homme et la machine.

Malgré une demande croissante, qui devrait être un signe de bonne santé, l'ergonomie de la CHM est en crise : il y a encore trop de malentendus sur le rôle que l'on veut lui faire jouer et sur sa nature même.

Le premier stade d'une étude ergonomique est l'identification des buts et des contraintes liées à l'opérateur, ainsi que celle des outils et des tâches à accomplir. Mais l'ergonomie de la CHM n'est pas monolithique et monothéorique. La description des contraintes peut se faire à différents niveaux et avec différents modèles théoriques. L'ergonomie est ainsi constituée de courants historiques différents qui entretiennent des ambiguïtés entre ingénieurs et ergonomes.

Tout a commencé par "l'hygiène et la sécurité du travail" dont l'objectif est de préserver la santé des opérateurs. On pensera par exemple aux agressions créées par les niveaux sonores trop élevés de certaines alarmes, ou aux polémiques des années 70 sur la nocivité supposée des écrans d'ordinateur. Il est clair que les problèmes d'hygiène existent, mais ils ne contraignent en pratique les systèmes que très faiblement, dans leurs aspects extrêmes.

Human factors

La deuxième étape, d'origine américaine, consiste à prendre en compte les *human factors* *. L'idée de base est simple : il faut élaborer des moyens de communication adaptés aux limites humaines ; par exemple ne pas utiliser des caractères trop petits, pour qu'ils restent lisibles sur l'écran.

L'ergonomie basée sur les *human factors* a une portée très générale. Elle va jusqu'à s'intéresser aux antagonismes possibles entre moyens de communication. Mais elle reste limitée, car elle est centrée exclusivement sur les données quantitatives des performances humaines : temps de réaction et nombre d'erreurs.

Aussi est-elle boudée par les ergonomes actuels car ses résultats sont globalement décevants. Un gain de quelques dixièmes de seconde peut avoir un sens statistique mais perdre toute signification si la pression temporelle n'est pas très grande. De la même façon, un dialogue sans erreur, mais alourdi (avec un métadialogue de contrôle de type "répétez", "confirmez", "validez", etc.) est souvent vécu comme inutile (exemple des distributeurs automatiques), voire handicapant dès lors qu'il entre en compétition avec d'autres activités qui doivent être conduites en même temps (cas du pilotage où le dialogue de confirmation oblige à un effort d'attention au détriment de la tâche principale). Enfin, il est classique de constater qu'un équipement fournissant une bonne performance lorsqu'il est testé isolément, s'avère inutilisable en contexte.

Il reste que les *human factors* sont précieux dans un stade précoce de développement des moyens de communications, ne serait-ce que parce qu'ils mettent en évidence les limites humaines que toute ergonomie plus sophistiquée devra respecter ; l'ergonomie des *human factors* n'est pas suffisante mais elle reste nécessaire.

L'ergonomie cognitive

La majorité des ergonomes français se reconnaissent dans un troisième courant : celui de l'ergonomie cognitive. En réaction avec le précédent courant, elle ne part pas des moyens de dialogue pour remonter vers la cohérence d'emploi, mais adopte la démarche inverse. Elle joue sur le qualitatif plus que sur le quantitatif et ne fait plus une distinction claire entre ergonomie de la tâche et ergonomie de la CHM.

L'enjeu du débat sur les courants théoriques de l'ergonomie est de fait celui de l'existence – ou non – d'une ergonomie de la CHM séparable de l'ergonomie de la tâche. La réponse est nuancée car il existe au moins deux niveaux d'action : le premier concerne le développement des outils de communication où une certaine indépendance d'étude est envisageable alors que lors de leur utilisation dans des cas concrets, les frontières tendent à s'estomper.

Les progrès récents de la couche communicante rendent la relation homme-machine beaucoup moins asymétrique, et contribuent encore plus à considérer la communication homme-machine comme un tout inséparable de la tâche. Le terme *dialogue* prend tout son sens sur des systèmes capables de se justifier, de contester, de conseiller l'usager humain. Une telle évolution a été rendue possible en dotant les systèmes d'un modèle de l'usager, actualisé au cours du dialogue, basé sur la tâche, qui permet de gérer intelligemment les répliques et le déroulement des tâches.

Le caractère finalisé du dialogue est essentiel. Plusieurs équipes travaillent en aéronautique sur des modèles encore plus élaborés qui permettent la reconnaissance automatique, en temps réel, des intentions des pilotes avec présentation pour confirmation sur le tableau de bord, de ce que la machine a compris.

Dans bien des cas les ingénieurs-chercheurs en CHM voudraient développer des moyens de communication sans risquer de se faire accuser de réaliser des études appliquées peu nobles. Ils sont donc réticents à observer des situations réelles et voudraient développer des concepts génériques. Les ergonomes de la CHM (les ergonomes tout court ?), chercheurs eux aussi, ne savent partir que de situations réelles dont ils inférent des propriétés. Les deux communautés doivent se rapprocher ; elles n'ont pas le choix ; leurs limitations respectives actuelles montrent qu'elles sont par nature complémentaires et que chacune détient un morceau de puzzle final.

* Il n'est pas possible de traduire ce terme par "facteur humain" dont le sens en français est beaucoup plus large.

Cabine de pilotage de l'Airbus A320 : par rapport aux avions de la génération précédente, on assiste à une simplification du tableau de bord © A. Ernoult.

René Amalberti, spécialiste de recherche du Service de santé des armées, Centre d'études et de recherches de médecine aérospatiale (CERMA), Département d'ergonomie aérospatiale, Base d'essais en vol, 91228 Brétigny-sur-Orge Cedex.

INDEX DES AUTEURS

FORUM DES RECHERCHES EN INFORMATIQUE 2 ET 3 JUIN 1993, ÉCOLE POLYTECHNIQUE

A la suite des rencontres "Recherche informatique-Industrie" organisées en juillet 1992 à Toulouse, les laboratoires et équipes participant aux Programmes de recherches coordonnées (PRC) du Ministère de la recherche et de l'espace, également constitués en Groupements de recherche du CNRS, vous invitent à participer au Forum des recherches en informatique, qui se tiendra les 2 et 3 juin prochains à l'Ecole polytechnique.

Cette manifestation est placée sous la tutelle du Ministère de la recherche et de l'espace, du CNRS, et de la Direction de la recherche et des études doctorales (DRED) du Ministère de l'Education nationale, avec le soutien de l'AFCET.

Lors de ce forum, des démonstrations et des exposés scientifiques et techniques donneront l'occasion aux visiteurs de faire le point sur l'état de la recherche en informatique en France, de nouer des contacts, d'amorcer des coopérations.

Les recherches exposées ne se limiteront pas aux équipes universitaires, mais s'étendront aux petites et moyennes entreprises et aux centres de recherche des grandes entreprises, centrées sur l'informatique, et ayant généralement des relations avec les équipes universitaires.

Les domaines couverts sont ceux des GDR et PRC :

Architectures nouvelles de machines,
Bases de données de 3ème génération,
Communication homme-machine,
Coopération, concurrence et communication,
Intelligence artificielle,
Mathématiques et informatique,
Programmation avancée et outils pour l'intelligence artificielle.

Pour tous renseignements :
Laboratoire d'informatique de l'Ecole polytechnique (LIX)
91128 Palaiseau Cedex
Tél. : (1) 69 33 40 73
Fax : (1) 69 33 30 14
Courrier électronique : fri@lix.polytechnique.fr

BON DE COMMANDE
à retourner à CNRS-Editions, 20-22, rue Saint-Amand, 75015 Paris

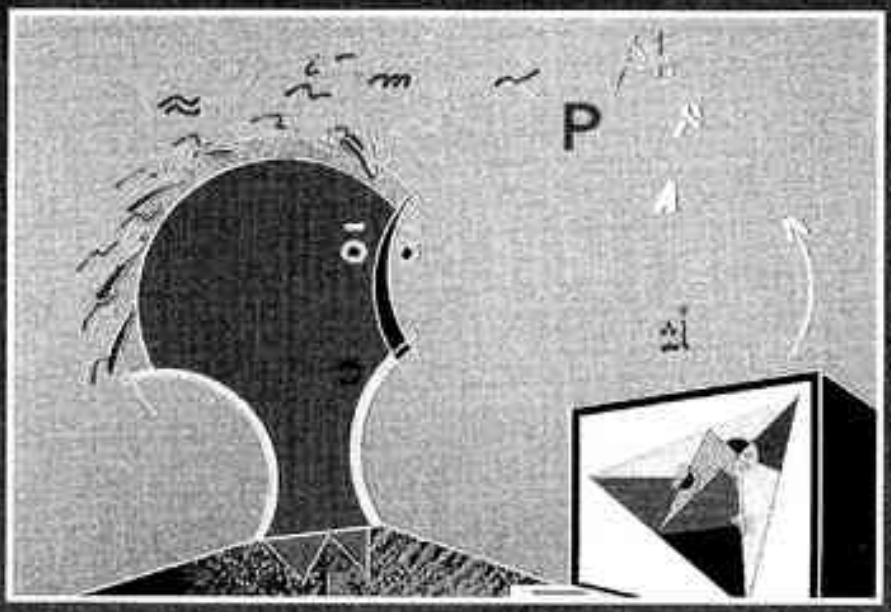
LE COURRIER DU CNRS
Dossiers scientifiques

ISBN	TITRE DE L'OUVRAGE	Prix	Qté	Total
04232-1	n°71: "La mécanique en 1988" (été1988)	50F		
04340-9	n°72: "Recherches sur l'environnement" (mai 1989)	50F		
04364-6	n°73: "Archéologie en France métropolitaine" (septembre 1989)	50F		
04396-4	n°74: "Le CNRS et les entreprises, la valorisation" (novembre 1989)	50F		
04446-4	n°75: "Les sciences du droit" (avril 1990)	50F		
04473-1	n°76: "La Terre: de l'observation à la modélisation" (juillet 1990)	50F		
04597-5	n°77: "Signaux et images" (juin 1991)	50F		
04664-5	n°78: "Politique et actions internationales au CNRS" (janvier 1992)	50F		
04689-0	n°78bis: "CNRS : international policy and programs" (janvier 1992)	50F		
04764-1	n°79 : "Sciences cognitives" (octobre 1992)	50F		

Port : France 15F - Etranger 20F
A partir de 5 numéros: franco de port

Total.....
Frais de port.....
Total.....

Nom.....Prénom.....
Adresse.....
Code postal.....Commune.....Pays.....
Ci-joint mon règlement de .....F à l'ordre de CNRS-Editions
Par : o chèque bancaire
o chèque postal
o mandat
o Carte Bleue: Je vous autorise à débiter mon compte Carte Bleue
N°.....Date de validité.....
Date.....Signature



CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE