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 Godel 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'Ecole 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 as 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.

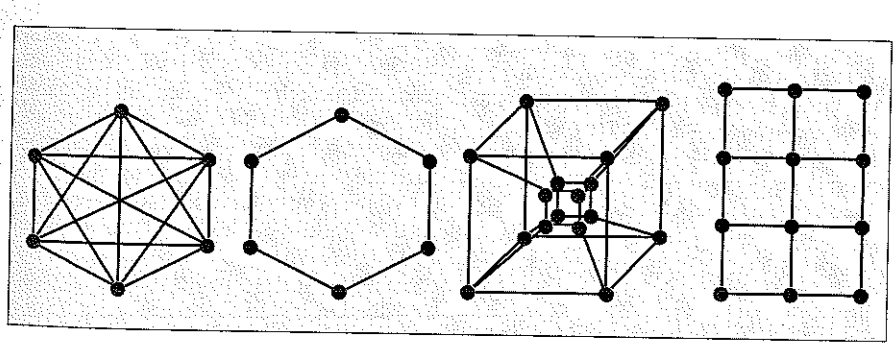

Fig. 1 - Un graphe complet, un anneau, un hypercube et une grille

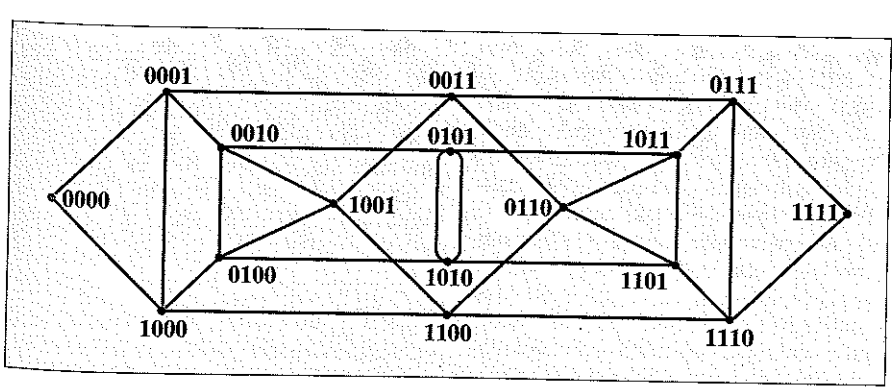

Fig. 2 - Un réseau de de Bruijn à 16 sommets

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.