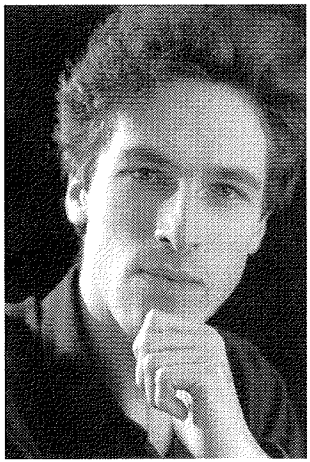
Sémantique Formelle des Langages de Programmations



M. Laurent Dami

Pour pouvoir présenter mes travaux, il me faut dire quelques mots sur la discipline intitulée *Sémantique Formelle des Langages de Programmations*, pas toujours connue des informaticiens, ni même des chercheurs en informatique. Cette discipline vise à décrire, par des modèles mathématiques rigoureux, les phénomènes survenant dans les programmes informatiques. Chacun a certainement vécu des expériences qui démontrent que l'informatique est, encore aujourd'hui, souvent hasardeuse, avec des hypothèses non vérifiées, des cas non prévus, des erreurs d'exécution. Les modèles mathématiques apportent d'une part un langage de communication totalement dépourvu d'ambiguïté, et d'autre part des techniques de preuves qui permettent d'acquérir la certitude que certaines propriétés sont vérifiées. Ces preuves peuvent dans certains cas être entièrement automatisées.

L'intérêt de la démarche est donc évident; ceci dit, il faut bien reconnaître que les résultats à l'heure actuelle sont relativement modestes. La théorie progresse, mais l'informatique de terrain progresse plus vite encore. La quasi-totalité des langages de programmation en usage aujourd'hui n'ont pas de modèle mathématique complet; de plus, dans les cas où un tel modèle existe, les propriétés que l'on réussit à prouver ne sont pas toujours les plus significatives. Pourtant des progrès très importants ont été accomplis, et permettent en retour de concevoir de nouveaux langages de programmation: ainsi il existe aujourd'hui dans le monde académique plusieurs langages entièrement formalisés, qui ont un intérêt certain, notamment pour l'enseignement. Malheureusement, à part quelques exceptions notoires, ces langages peinent encore à s'imposer hors des cercles universitaires.

Même dans les cas où la modélisation mathématique n'est pas complète, la sémantique formelle permet néanmoins d'établir certains résultats partiels qui ne sont pas dénués d'intérêt. L'INRIA en France a publié récemment une étude exemplaire à ce propos. On se souvient de l'échec retentissant de la fusée Ariane 5 en 1996, lors duquel plusieurs centaines de millions de francs sont partis en fumée... à cause d'une erreur de programmation! Or les outils actuels d'analyse automatique de programmes, inspirés de ces fameux modèles mathématiques, auraient permis de détecter l'erreur en question.

Contrairement à d'autres secteurs de l'informatique, les travaux en sémantique formelle ont une tradition européenne très forte, avec de nombreux représentants éminents en Italie, en France, en Grande-Bretagne, aux Pays-Bas. Dans certains pays, cette discipline fait même l'objet d'un enseignement de premier cycle. En revanche elle est quasiment inexistante en Suisse, pour des raisons historiques qui m'échappent. Il se peut que l'abondance de moyens, telle que nous l'avons vécue dans ce pays pendant de nombreuses années, ait encouragé les recherches sur de gros équipements, au détriment des travaux ne nécessitant essentiellement que papier et crayon.

Je vais maintenant tenter d'exposer la teneur de mes travaux. Ceux-ci s'appuyent sur un formalisme absolument central pour les recherches en sémantique formelle, et qui a été inventé dans les années 1930 déjà, soit bien avant les premiers ordinateurs: il s'agit du *lambda-calcul* du logicien Alonzo Church. La motivation de Church était de construire les fondements des mathématiques sur la notion de *calculabilité*. Son formalisme permet d'exprimer cette notion avec élégance et avec une grande économie de moyens. Ce n'est que bien plus tard que l'on a constaté que le lambda-calcul était suffisamment expressif pour décrire une grande variété de phénomènes informatiques, et que son petit nombre de concepts en faisait un outil idéal pour les travaux théoriques. Aujourd'hui encore de nombreux travaux non seulement utilisent ce formalisme, mais le prennent comme objet d'étude. On cherche par exemple à établir dans quelles conditions deux programmes exprimés en lambda-calcul peuvent être considérés comme équivalents, c'est-à-dire qu'ils accomplissent la même tâche. Parmi ceux qui sont équivalents, on cherche ensuite à caractériser celui qui est le plus économe en ressources. Enfin, on développe des techniques d'analyse qui permettent par exemple de détecter et d'éliminer certains types d'erreurs.

La majorité des résultats dans le lambda-calcul concernent des expressions dites *fermées*, qu'on peut comparer à des boîtes noires, dans lesquelles on injecte certaines données, et qui produisent certains résultats en retour. Les programmes classiques correspondent très bien à ce modèle, puisqu'ils constituent des entités complètes et immuables,

dont le comportement ne dépend plus que des données qu'on leur injecte. Pourtant, depuis plusieurs années, la programmation a évolué vers un modèle beaucoup plus dynamique. Les programmes récents ne sont plus des entités immuables et rigides, mais des assemblages de composants, qui peuvent se modifier avec le temps. Aujourd'hui, pour adapter un programme, on peut souvent se contenter de remplacer l'un de ses composants, qui sera automatiquement reconnecté aux autres composants, sans aucune intervention du programmeur. Mieux, l'assemblage des composants peut même survenir en cours d'exécution du programme. Par exemple un simple programme de traitement de texte est aujourd'hui appelé à collaborer avec des bases de données, avec des programmes de traitement d'image, à utiliser Internet pour rapatrier des informations, et ceci sans que l'on puisse prédire la totalité des interactions au moment où le traitement de texte est démarré; en conséquence la frontière de ce programme de traitement de texte devient relativement floue, ce qui complique d'autant son étude théorique. Le lambda-calcul classique est mal équipé pour traiter de telles situations, car toute modification sur l'une de ces fameuses boîtes implique de reprendre l'ensemble des réseaux de câblage qui l'environnent.

Ma contribution a consisté, dans un premier temps, à introduire dans le lambda-calcul un mécanisme de nommage qui permette d'exprimer la notion d'assemblage dynamique de composants. Pour parler en images, les boîtes noires n'ont plus un nombre fixe de tuyaux d'entrée ou de sortie; elles ont un nombre variable de tuyaux, étiquetés par des noms, ce qui leur permet, en cours d'exécution, de se connecter avec d'autres boîtes pour accomplir un travail commun. De plus, s'il s'avère nécessaire de modifier ou d'améliorer une boîte, on peut le faire en lui rajoutant des câbles, mais sans avoir à toucher aux connexions existantes.

La conséquence logique de cette proposition a été de remettre en cause dans un deuxième temps la notion usuelle d'équivalence. En effet, avec de telles boîtes, il est peu intéressant de déterminer si l'une est totalement équivalente à une autre, car alors on perdrait une bonne partie de la flexibilité recherchée. En revanche, il est important de

pouvoir déterminer si une boîte peut en remplacer une autre en toutes circonstances: intuitivement, on comprend bien que si une boîte est remplacée par une nouvelle version améliorée, il est important de garantir que l'ensemble du circuit fonctionnera toujours correctement. La relation entre l'ancienne et la nouvelle boîte n'est donc plus symétrique mais asymétrique. C'est pourquoi j'ai proposé de réviser la méthodologie traditionnelle en sémantique, au profit de cette nouvelle relation baptisée *subsumption*.

Il serait difficile d'aller plus avant dans la description de mes travaux sans entrer dans des détails techniques fastidieux pour les profanes. En revanche il vaut peut-être la peine de consacrer quelques lignes au parcours qui m'a amené à entreprendre de telles recherches. En effet, ni ma formation initiale en sciences économiques et sociales, ni mes premiers travaux en animation, multimédia et programmation orientée-objets, ne me prédestinaient a priori au lambda-calcul. C'est progressivement que j'ai délaissé ces domaines, pourtant à la mode et éminemment porteurs, pour me tourner vers la théorie. L'intérêt personnel a bien entendu été un moteur essentiel dans cette démarche, en partie sous l'influence du best-seller de Douglas Hofstadter, *Gödel, Escher, Bach*, qui constitue une magnifique introduction à l'informatique théorique. Il est cependant un autre facteur, qui est celui du souci de la position de l'université vis-à-vis de l'industrie. Face à la vitesse d'évolution du monde informatique, face aux moyens considérables dégagés dans le privé pour tous les domaines à fort potentiel commercial, le danger est grand pour le chercheur académique de se retrouver à courir après le train, sans avoir la possibilité de dégager une contribution significative et qui ait un minimum de durabilité. En comparaison, il est fascinant de constater que les travaux initiaux liés au lambda-calcul sont encore cités près de 70 ans après leur parution! Cette remarque me paraît pertinente dans la mesure où une pression politique toujours plus forte veut encourager les rapprochements entre la recherche et l'industrie; pour ma part, je crains qu'une évolution trop marquée dans ce sens fragilise fortement les chercheurs, en tout cas en informatique, en les orientant de préférence vers des travaux qui risquent d'être très rapidement périmés.

Pour conclure, et pour rester dans un registre personnel, je me dois d'exposer brièvement les raisons qui m'ont poussé à quitter le monde de la recherche... quelques mois à peine avant de recevoir le Prix Latsis! Je travaille en effet aujourd'hui comme conseiller en systèmes d'information au sein de l'administration genevoise, où je m'occupe de problèmes qui n'ont plus grand-chose de théorique, mais représentent des enjeux très concrets, comme par exemple le passage de l'an 2000. Ce nouveau cadre m'a permis d'acquérir en peu de temps de nombreuses dimensions qui faisaient défaut au chercheur, telles que la prise de conscience des aspects humains, organisationnels ou politiques liés aux systèmes d'information, et je ne regrette donc en rien d'avoir ainsi changé d'orientation. Néanmoins il est indéniable qu'une telle décision a été motivée en bonne partie par un souci économique. Ceci ne fait que confirmer un constat déjà établi par le Fonds National de la Recherche Scientifique, et relayé dans la presse: le statut des chercheurs intermédiaires en Suisse est extrêmement précaire. A partir d'un certain âge et d'un certain niveau de responsabilité familiale, une telle fragilité devient inacceptable, et il ne reste plus dès lors que l'exode... ou la reconversion, ce qui dans les deux cas représente une perte pour la recherche suisse. Pour ma part, j'ai préféré renoncer à la mobilité géographique, en dépit du prestige dont elle jouit dans les milieux économiques et académiques, car elle me semble moins importante à l'ère d'Internet: les moyens de communication électronique et les programmes internationaux de recherche m'ont permis de constituer un réseau de contact internationaux de qualité sans délaisser mes attaches genevoises. En revanche, la mobilité intellectuelle, à savoir le passage d'un domaine technique extrêmement pointu à des considérations beaucoup plus pragmatiques, a constitué pour moi un défi d'un grand intérêt, qui continue de me passionner.