New Challenges for Theoretical Computer Science

Maurice Nivat
Liafa,Universite Paris7-Denis Diderot

Abstract Recent developments of both practice and theory and the emergence of new needs of actual computation or software engineering allow us to indicate a number of areas in which research is likely to be extremely active and productive in the next decade. These are obviously also difficult areas which requirenew concepts new methods and sophisticted tools of all kind many of them still to be built.

L'informatique théorique s'est constituée dans les années 60 pour combler des lacunes dans la connaissance de certains objets mathématiques que l'informatique naissante amenait à manipuler de façon cruciale : le meilleur exemple en est la théorie des grammaires et langages formels qui était nécessaire à la mise au point de compilateurs pour les langages de programmation que l'on appelait symboliques à cette époque, Fortran, Algol et bien d'autres. Elle a d'ailleurs parfaitement rempli son objectif, analyseurs lexicaux et syntaxiques indispensables à cette compilation étaient très bien maîtrisés dès le début des années 1970. En même temps était née et avait pris son essor ce que l'on peut appeler la théorie des langages et des automates, qui après l'inévitable période de cafouillage initial avait posé les bonnes définitions, qui durent encore, sont universellement utilisées et enseignées, et propose des conjectures difficiles, de celles qui ne se résolvent pas en un jour ni en un an, dans la mesure où leur résolution exige une connaissance approfondie des structures mises en jeu, donc beaucoup de travail.

Les autres chapitres de l'informatique théorique se sont constitués peu après, comme la sémantique formelle, née de la problématique de la compilation automatique qui nécessitait la description précise, non ambigüe, de la sémantique d'un programme ou d'une spécification de programme, et ouvrait la voie à toute l'ingénierie du logiciel moderne. Quant à l'algorithmique, elle se développait sur deux plans; l'algorithmique" naïve "consistant à mesurer les performances des algorithmes en terme de nombre d'opérations à effectuer, et à la place en mémoire indispensable pour l'obtention d'un certain résultat (dans les premiers temps, les problèmes de tris et de recherche en table occupaient une place prépondérante dans cette problematique). L'algorithmique théorique, pas vraiment nouvelle puisqu'elle était pratiquée depuis longtemps par des logiciens de la calculabilité, trouvait son chemin de Damas avec le fameux problème *P=NP*? qui a résisté jusqu'à présent à toutes les tentatives de résolution, mais suscite une masse impressionnante de travaux ayant eu des conséquences très directes sur la façon de traiter et résoudre les innombrables problèmes *NP* complets que soulévent

même la vie la plus quotidienne (emploi du temps, ordonnancement de tâches, synchronisation, etc).

Les informaticiens, qui sont un peu masochistes pris collectivement, ont pour habitude de se poser très souvent la question de l'avenir de leur discipline et de sa place dans le concert des sciences, aidés en cela par les scientifiques d'autres disciplines qui bien souvent ne comprennent pas la problèmatique de l'informatique réduite à leurs yeux à une simple technique ancillaire : n'y a-t-il pas de par le monde des milliers de gens qui se servent avec bonheur d'ordinateurs pour résoudre leurs problèmes sans jamais avoir appris fussent les rudiments les plus élémentaires de l'informatique théorique? Une question lancinante pour les informaticiens théoriciens est le raport des mathématiques et de l'informatique : leur discipline est-elle un simple chapitre des mathématiques? Vont-ils disparaître submergés par les gros bataillons de mathématiciens armés de toute leur science, dès que ceux-ci vont s'attaquer aux problèmes qui leur sont chers?

Je souhaite ici apporter quelques éléments de réponse à ces questions et réconforter les informaticiens théoriciens qui redoutent d'être bientôt mis au chômage, après avoir été ridiculisés, par les mathématiciens qui vont résoudre les problèmes sur lesquels ils suent sang et eau en deux coups de cuiller à pot. Cela n'arrivera sûrement pas, cela n'a aucune chance d'arriver. D'abord parce que nos problèmes sont innombrables et s'enrichissent chaque jour de questions nouvelles liées à la pratique de l'informatique et exigeant de nouvelles avancées théoriques dont tout porte à croire qu'elles prendront des années. Voici quelques exemples de ces questions nouvelles de nature à occuper de très nombreux très bons chercheurs pendant des années à venir, j'en propose appartenant à tous les domaines de l'Informatique théorique : les automates finis se sont, dans les années qui viennent de s'écouler, révélés être la bonne structure pour étudier des systèmes complexes mettant en jeu plusieurs processeurs communiquant entre eux et conspirant à l'obtention d'un certain résultat, qui très souvent est seulement la bonne marche d'un systeme mécanique ou physique sophistiqué, tel un avion ou un reacteur nucléaire. L'écriture, la mise au point et la validation des logiciels de commande de tels systèmes amène à considérer d'énormes automates finis qui décrivent de fait l'incroyable complexité combinatoire engendrée par la multiplicité des paramètres et des situations dans lesquelles peuvent se trouver les dits systèmes. L'analyse des dits automates est un problème pratique fondamental et théoriquement extrêmement difficile : comme il est d'usage quand un besoin pratique précède la connaissance théorique, des méthodes ont été mises au point, sont vendues et utilisées, dont on ne connaît véritablement pas le domaine de validité et que l'on ne comprend que très imparfaitement. Ce n'est pas tout : dans ce que l'on appelle désormais système hybride, coexistent une partie de contrôle régie par des automates finis et des équations d'évolution continues, différentielles ou aux dérivées partielles, et la considération des systèmes hybrides amène à repenser toute la théorie du contrôle telle qu'elle a été élaborée pendant des décennies par les automaticiens : beau champ d'activité !

L'algorithmique que j'ai appelée naïve acommencé par évaluer des nombres d'opérations à effectuer dans le cas le plus défavorable, puis évidemment des nombres moyens d'opérations en supposant les cas possibles équidistribués. Déjà elle est devenue moins naïve ce faisant, et a été obligée de développer des techniques mathématiques, par exemple d'analyse asymptotique, assez sophistiquées; là comme ailleurs , bien sûr, ces techniques ne sont pas entièrement nouvelles et se sont développées à partir de résultats et de méthodes antérieurement mis à jour par des mathématiciens, mais il est aussi évident que les besoins très précis de l'analyse d'algorithmes ont amené des développements nouvaux auxquels, si j'ose dire, les mathématiciens n'avaient aucune raison de s'intéresser ni de penser. Plus récemment, l'algorithmique a encore fait un pas en avant considérable avec l'avènement de l'algorithmique que j'appelerai "randomisée" : tests aléatoires de primalité de grands entiers, ouvrant la voie à une méthode très générale encore très peu explorée. Le fameux problème du seuil pour la satisfiabilité consiste à calculer des probabilités, qui ont été d'abord observées par des praticiens : ce problème a déjà suscité un nombre impressionnant de travaux remarquables, originaux et novateurs, et de plus il laisse présager des avancées de même nature dans nombre de problèmes a priori difficiles (NP) et très répandus (sac à dos, emplois du temps, etc).

Le logiciel, dont les avancées sont nombreuses, se trouve confronté au problème amusant, dit de l'an 2000, problème soulevé par le fait que d'innombrables programmes dans le monde entier n'ont prévu que deux chiffres pour indiquer des dates, et que dans trois ans il en faudra davantage. Corriger à la main tous ces programmes pour tenir compte de cette nouvelle exigence est simplement impossible, tous les programmeurs de la planète travaillant d'arrache-pied24 heures sur 24 à ce seul problème pendant les 3 ans qui viennent n'y suffiraient pas! D'où la seule solution qui est de la réingénierie des logiciels existants, consistant à abstraire et représenter dans une structure adéquate, toute la sémantique des logiciels en question, puis de leur réécriture automatique incorporant les modifications voulues, mais aussi sous une forme qui prévoit la maintenance et l'évolution future des dits logiciels. Assez fascinant problème sur lequel de nombreux chercheurs et ingénieurs ont commencé à se pencher, et qui fait appel à vraiment toutes les connaissances accumulées jusqu'à présent en matière de programmation , de sémantique, de représentations de données ou de connaissances : gageons que le problème de l'an 2000 va se traduire par de très nombreux et remarquables résultats, aussi bien à la connaissance intime des phénomènes de programmation qu'à l'outillage pour produire et transformer du logiciel!

Sur une registre moins industriel la programmation par contraintes, technique développée depuis plusieurs années, rentre dans une phase tout à fait passionante, dans la mesure où les systèmes se mettent à incorporer toutes les heuristiques imaginées pour résoudre des problèmes difficiles dans la plupart des cas, et donc à produire des programmes ayant une efficacité comparable aux meilleurs produits à la main et pourris d'astuces. On peut imaginer désormais des systèmes automatiques, non plus

seulement de production de programmes implémentant un algorithme donné, mais d'algorithmes adaptés à des situations concrètes, correspondants à diverses formes de contraintes et choisissant, en fonction des résultats d'une analyse de ces contraintes, l'heuristique ayant le plus de chance de conduire à un resultat exact ou à une très bonne approximation.

Je pourrais multiplier les exemples : je crois que ceux-là suffisent pour se convaincre que l'informatique même la plus théorique, ne manque pas de problèmes à court, moyen et long terme, qui d'une part sont en prise directe sur une pratique réelle et répondent à des besoins de nombreux secteurs de l'activité humaine, et d'autre part appellent des éeveloppements théoriques difficiles qui devront passer par des mathématiques encore à imaginer ou à construire. J'ai plutôt tendance à penser que c'est maintenant que l'informatique théorique va devenir tout à fait intéressante, et dépasser les frontières des problèmes assez immédiats de définition des structures de base : que la plupart des problèmes évoqués ci-dessus exigent, à l'évidence, une collaboration étroite entre informaticiens et autres scientifiques mathématiciens, statisticiens, automaticiens ou roboticiens, physiciens ou biologistes (pour des problèmes non évoqués ci-dessus mais tout aussi passionants, ceux qui sont liés au génome comme ceux qui sont liés à toute espèce de systèmes dynamiques discrets ou d'états instables de la matière) n'est certainement pas quelque chose qu'il faille redouter mais bien plutôt l'annonce que collaborant avec d'autres chapitres de sciences sur des problèmes très difficiles et importants, les informaticiens trouveront enfin effectivement toute la place qui leur revient dans le développement de la Science.