



Leslie Lamport : l'importance de la pensée mathématique pour l'informatique

Stephan Merz¹, Anca Muscholl²

En mars dernier, l'ACM a annoncé que le prix Turing 2013 était attribué à Leslie Lamport pour ses « contributions fondamentales à la théorie et à la pratique des systèmes répartis et parallèles, notamment l'invention de concepts comme la causalité et les horloges logiques, de sûreté et de vivacité, les machines à états répliqués et la consistance séquentielle »³.

L'original en anglais de cette citation, ainsi que plusieurs développements, se trouvent sur les sites :

<http://www.acm.org/news/featured/awards/turing-award-2013>

et

http://amturing.acm.org/award_winners/lamport_1205376.cfm

L'article le plus cité de Leslie Lamport, qui est également un des plus cités de l'histoire de l'informatique, s'intitule « Time, Clocks, and the Ordering of Events in a Distributed System », et pose les fondements du calcul réparti. On le trouve à :

<http://amturing.acm.org/p558-lamport.pdf>

1. Directeur de recherche au Centre de recherche Inria Nancy - Grand Est, membre du LORIA.

2. Professeure à l'université de Bordeaux, membre du LaBRI et du Conseil scientifique de la SIF.

3. Voir également l'article paru dans le numéro 3 de 1024, <http://www.societe-informatique-de-france.fr/wp-content/uploads/2014/05/1024-3-lamport.pdf>.

La rubrique REGARDS de ce numéro présente les opinions de Lamport sur l'importance de la pensée mathématique pour l'informatique, qui motivent ses travaux actuels sur la spécification formelle et la vérification de programmes parallèles.

Enfin, n'oublions pas son rôle essentiel dans le développement du logiciel L^AT_EX dont la communauté scientifique ne saurait se passer.

Marie-Claude Gaudel

En janvier 2013 Leslie Lamport⁴ est invité à donner son opinion sur les pratiques courantes en matière de programmation au site grand public www.wired.com. Son article *Pourquoi nous devrions construire des logiciels comme nous construisons des maisons*⁵ suggère que les programmeurs devraient consacrer un peu de leur temps à décrire, ne serait-ce qu'en langue courante, ce qui est attendu d'un morceau de code, et ceci avant de le produire. « Réfléchir ne garantit pas que nous ne commettrons pas d'erreurs. Ne pas réfléchir garantit que nous en allons commettre. » Cet article suscite l'incompréhension de beaucoup de lecteurs qui jugent notamment que les contraintes de l'industrie moderne du logiciel ne laissent pas de temps pour rédiger des spécifications qui deviendront d'ailleurs très vite obsolètes à cause des évolutions incessantes des besoins.

Lamport n'en est pas à son coup d'essai : dès 1977, inspiré par les travaux de Floyd et de Hoare, il avait réfléchi sur la perception de la programmation dans un monde moderne qui en est de plus en plus dépendant. Dans une présentation à l'Université de Kiel (Allemagne) en 2003, il compare les programmes modernes à des systèmes biologiques, dont la complexité les rend difficilement compréhensibles. Il y prône une approche rigoureuse selon laquelle un programme est un objet mathématique complexe mais qui reste à la portée d'une analyse logique. D'après Lamport, la logique doit intervenir dans la conception de programmes et le raisonnement sur leurs propriétés, et elle ne doit pas être réservée aux seuls spécialistes, mais ses concepts fondamentaux doivent être compris par les développeurs et ingénieurs : « La pensée mathématique est quelque chose que les ingénieurs devraient apprendre à l'université plutôt que d'y être confrontés seulement en milieu professionnel. Ils peuvent apprendre au travail les spécificités d'un langage de programmation, mais la façon de penser doit être enseignée par l'université. De nos jours, les étudiants américains peuvent apprendre C++, Java et les détails d'Unix. Mais très peu d'universités américaines offrent des cours qui abordent la conception des systèmes réels sous un angle mathématique et logique. Les universités sont appelées à faire plus d'efforts dans cette direction. »

4. <http://www.lamport.org/>

5. <http://www.wired.com/2013/01/code-bugs-programming-why-we-need-specs/>

Les contributions principales de Lamport concernent la conception d'algorithmes répartis, c'est-à-dire exécutés par des nœuds indépendants dans un réseau d'ordinateurs. Dans ce domaine, selon le dicton bien connu de Lamport, « la panne d'un ordinateur dont vous ignoriez même l'existence peut faire capoter le calcul ». Les apports de Lamport ont été profonds et ont marqué le domaine, que ce soit par la formalisation de notions clés comme celle de tolérance aux pannes (problème des généraux byzantins) ou par la conception d'algorithmes. Par exemple, l'algorithme *Paxos* qui vise à garantir un consensus parmi des nœuds répartis est à la base de nombreux services Internet et Cloud modernes, comme la gestion de documents modifiables par des utilisateurs distants ou les services d'achats en ligne.

Énoncer précisément les fonctionnalités attendues d'un algorithme réparti et décrire sans ambiguïté son opération, ainsi que les hypothèses (mode de communication, pannes...) sur lesquelles repose son fonctionnement est fondamental pour concevoir des systèmes fiables. Aussi existe-t-il un sous-domaine de l'informatique qui conçoit des langages et formalismes pour la spécification d'algorithmes et de systèmes informatiques et pour raisonner sur leurs propriétés. Fidèle à sa formation initiale de mathématicien, Lamport se montre sceptique à l'égard de formalismes ad-hoc et prône l'utilisation, autant que possible, des notations et concepts mathématiques habituels dont il considère qu'ils s'appliquent parfaitement aux systèmes informatiques. De ces idées est né le langage logique de spécification TLA^+ , dont le but est de rendre la conception et le raisonnement formel autour des systèmes concurrents complexes plus abordable.

Quelle conclusion tirer sur les contributions de Leslie Lamport, qui a marqué l'informatique moderne par son approche originale, à la fois scientifique et pourvue d'un sens prononcé pour les problèmes réels ? Un des messages clés de l'activité de Lamport est que l'informatique, et en particulier la programmation, doit se nourrir d'une conception rigoureuse, mathématique, qui font de l'informatique la science que nous aimons, développons et enseignons : « Lorsque des gens qui ne pensent pas de façon logique conçoivent de grands systèmes, ces systèmes deviennent incompréhensibles. Et alors nous commençons à regarder ces systèmes comme des systèmes biologiques. Les systèmes biologiques étant trop complexes, il semble très naturel de penser aussi que les systèmes informatiques sont trop complexes pour être maîtrisés. Nous ne devons pas accepter cette situation — « nous » signifiant les informaticiens en première ligne, mais aussi les utilisateurs de l'informatique. »