Xavier Leroy obtient le prix Milner

Gérard Berry [1]

Le 24 novembre 2016, dans les locaux de la Royal Society à Londres, Xavier Leroy recevra le Milner Award [2]. *C'est le plus grand prix européen en informatique, décerné conjointement par la Royal Society, l'Académie des sciences, et l'Académie allemande Leopoldina en l'honneur du grand informaticien britannique Robin Milner (prix Turing 1992). Cette récompense lui est décernée pour ses contributions théoriques et pratiques au domaine de la programmation, avec en particulier le développement des langages Caml puis OCaml [3], puis celui du compilateur C mathématiquement certifié CompCert [4] au sein du projet Inria Gallium [5] qu'il dirige.*

Le développement de langages de programmation efficaces et sûrs reste un des points durs de l'informatique. Même si plusieurs centaines de langages et plusieurs milliers de dialectes ont été introduits depuis les années 1950, bien peu possèdent en même temps les deux caractéristiques précitées. Caml puis OCaml sont de ceux-là. Caml est une extension du langage fonctionnel ML (*Meta-Language*) développé par Robin Milner et son équipe de l'université d'Edimbourg au début des années 1980, originellement pour construire les preuves de programmes dans le système pionnier de vérification de programmes LCF (*Logic for Computable Functions*). Par rapport à

1. Professeur au Collège de France, membre de l'Académie des sciences, http://www-sop.inria.fr/members/Gerard.Berry/

2. https://royalsociety.org/science-events-and-lectures/2016/11/milner-award-lecture/

3. http://ocaml.org/

4. http://compcert.inria.fr/

5. http://gallium.inria.fr/

ses concurrents de l'époque, ML se distinguait par son système de type polymorphe (i.e., tel qu'une structure de données ou une fonction peuvent prendre n'importe quel type d'arguments, comme par exemple une liste homogène d'éléments d'un même type quelconque). Ce typage rigoureux des programmes permettait une programmation à la fois très souple et très rigoureuse en détectant un maximum d'erreurs dès la compilation. Tous les systèmes de types des langages fonctionnels modernes en sont issus.

Gérard Huet initia en 1982 le développement d'un noyau ML extrait de LCF et doté d'un compilateur autonome, en collaboration avec Larry Paulson de l'université de Cambridge. Le langage se développa au sein de l'équipe Formel, au bâtiment 8 du centre IRIA de Rocquencourt (devenu plus tard Inria). ML devint Caml au milieu des années 1980, quand Guy Cousineau et Michel Mauny y utilisèrent la CAM comme machine virtuelle intermédiaire ; CAM abrège *Categorical Abstract Machine*, une machine abstraite inventée par Pierre-Louis Curien pour évaluer les langages fonctionnels. Mais l'efficacité d'un système multicouche laissait à désirer.

Au milieu des années 1990, Xavier Leroy, alors thésard dans l'équipe, reprit le problème à zéro. En collaboration avec son collègue Damien Doligez, il réalisa une implémentation légère mais très efficace du langage Caml en C, appelée Caml-Light. Ce langage et cette implémentation commencèrent à se répandre, notamment en France pour l'enseignement de la programmation. Ensuite, avec François Pottier, Xavier Leroy dota le langage d'un système de modules paramétriques très expressif, autorisant le développement de logiciels structurés de très grande complexité. Enfin, Jérôme Vouillon et Didier Rémy complétèrent le langage avec une couche de constructions objets. OCaml était né. Il servit de langage d'implémentation de l'assistant de preuves Coq ainsi que de nombreux autres logiciels académiques, et acquit progressivement une pertinence industrielle dans le domaine de la programmation certifiée. Esterel Technologies utilisa par exemple OCaml pour la programmation du langage et du compilateur SCADE 6, langage formel qui sert à écrire les lois de pilotage et les autres fonctions automatisées essentielles à de nombreux avions et autres systèmes industriels. Dans de tels systèmes, l'informatique doit être certifiée au plus haut niveau : le compilateur SCADE 6 est effectivement certifiable selon la norme DO-178C pour le logiciel avionique, la plus contraignante en matière de développement logiciel. Progressivement, OCaml acquit la renommée de son slogan « *The language of choice for the discriminating hacker* » et nombre d'entreprises de pointe le choisissent aujourd'hui comme support de développement logiciel, notamment pour les applications bancaires ou de gestion de réseaux sociaux. La page Web *Companies using OCaml* [6] en témoigne abondamment.

Il faut aussi souligner que Xavier Leroy n'est pas qu'un implémenteur exceptionnel : ses contributions théoriques sur les théories des types et modules, présentées

6. `http://ocaml.org/learn/companies.html`

par exemple dans les conférences POPL (*Principles of Programming Languages*), dont il est un des piliers, ou dans d'autres conférences et journaux de premier plan, ont eu un impact considérable.

CompCert, l'autre grande contribution de Xavier Leroy et de son équipe, est une aventure encore unique sur le plan mondial. Il s'est agi de réaliser un compilateur du langage C qui soit non plus certifié seulement pour la qualité des méthodes de développement et de test comme dans la norme DO-178C, mais bel et bien prouvé mathématiquement correct et donc vraiment garanti sans bugs. Un tel développement est une course de haies : il faut d'abord choisir un sous-ensemble approprié de C, qui est loin d'être un langage bien défini. Le sous-ensemble C-light que compile CompCert correspond bien aux besoins du logiciel industriel embarqué dans les systèmes physiques. Il faut ensuite choisir des sous-ensembles des langages machine des processeurs (Intel, Motorola, Mips, etc.) qui soient également formellement définissables. Une fois ces sous-ensembles choisis, il faut définir leur sémantique mathématique de façon appropriée et directement implémentable dans un assistant de vérification. Enfin, il faut démontrer un théorème du type « pour tout programme P écrit en C et toute machine cible M, si CompCert produit un code machine PM pour M à partir de P, alors exécuter PM sur M donne pour toute entrée le même résultat qu'exécuter M dans la sémantique de C ».

Cette preuve, très complexe, ne peut pas se faire directement. Pour CompCert, elle passe par 11 étapes correspondant chacune à un nouveau niveau de langage intermédiaire. Chaque étape est prouvée correcte à l'aide de l'assistant de preuves Coq, développé en France et en OCaml, qui est devenu la référence du domaine. Mieux, la plus grande partie du code OCaml du compilateur obtenu est automatiquement extrait de la preuve. Quelquefois, par exemple pour certaines phases complexes d'optimisation, CompCert fait appel à des programmes externes non vérifiés ; mais, à chaque fois, il vérifie formellement que le résultat fourni par ces programmes est bien conforme. Enfin, ce qui est essentiel pour les applications, CompCert n'est pas un compilateur jouet. Il compile les programmes C assez rapidement (en gros deux fois plus lentement qu'un compilateur standard), et, grâce à des optimisations bien choisies et formellement prouvées correctes, le code qu'il génère est du même niveau d'efficacité que celui généré par les compilateurs standards avec le niveau d'optimisation utilisé en embarqué, souvent peu agressif car des optimisations trop complexes peuvent introduire des bugs sournois.

L'importance de CompCert devient claire quand on regarde d'autres projets comme l'outil Csmith [7] développé à l'université d'Utah, qui, à l'aide de techniques de génération aléatoire, a permis d'engendrer automatiquement un million de programmes C destinés à casser les compilateurs. Le résultat a été étonnant : Csmith a permis de détecter des centaines de bugs dans les principaux compilateurs C, mais

7. https://embed.cs.utah.edu/csmith/

aucun dans CompCert (en fait deux au début, mais pas dans les parties bien définies de C, et ces bugs ont ensuite été corrigés).

Être parvenu à construire CompCert est clairement un exploit scientifique et technique de première grandeur et impensable il y encore 15 ans. Et cet exploit en entraînera d'autres, par exemple dans le projet américain DeepSpec [8], un grand projet coopératif récent conduit par un consortium de 14 universités visant à généraliser la preuve mathématique à une grande variété de logiciels en tous genres. En résumé, les apports de Xavier Leroy, autant théoriques que pratiques, constituent une avancée d'une importance considérable dans le domaine de la programmation sûre des systèmes modernes. Ils ont des applications importantes dans la recherche publique et privée et dans de plus en plus de domaines industriels, où OCaml est considéré comme un des meilleurs langages et CompCert le sera certainement comme un jalon essentiel vers des systèmes plus sûrs ; OCaml sert aussi depuis longtemps de base à beaucoup d'enseignements.

Xavier Leroy a déjà reçu de nombreux prix, mais le prix Milner le distingue d'autant plus particulièrement que Robin Milner peut être considéré comme le fondateur de sa discipline. Si vous ne pouvez pas vous rendre à Londres, son exposé sera disponible sur le Web :

`https://royalsociety.org/science-events-and-lectures/2016/11/milner-award-lecture/`

8. `http://www.deepspec.org/`