COMMUNIQUE DE PRESSE – 29 OCTOBRE 2018

Création d'une chaire *Sciences du logiciel* au Collège de France

Xavier Leroy, nommé titulaire, prononcera sa leçon inaugurale

le jeudi 15 novembre 2018, à 18h00

Après avoir ouvert en 2009 la chaire annuelle *Informatique et sciences numériques*, puis, en 2012, la chaire *Algorithmes, machines et langages* (confiée à Gérard Berry), et en 2017, la chaire *Sciences des données* (confiée à Stéphane Mallat), l'Assemblée du Collège de France a établi une chaire entièrement consacrée aux s*ciences du logiciel*. Le Pr **Xavier LEROY** a été nommé titulaire de cette nouvelle chaire.

> *« Le logiciel sans bug est possible. Plus précisément, il n'existe pas d'impossibilité fondamentale interdisant ce résultat ».*
>
> Xavier LEROY

Langages de programmation, fiabilité et sécurité informatique

Pour le grand public, la notion de logiciel est souvent synonyme de "plantage" et de failles de sécurité. Pourtant, il existe des systèmes logiciels critiques qui atteignent des niveaux de fiabilité extraordinaires. Par exemple, des systèmes de commande de vol électriques (*fly-by-wire*) impliquant une quantité considérable de logiciels sont utilisés dans les avions commerciaux depuis près de 40 ans sans qu'aucun "bug" ne vienne causer d'incident.

Que faut-il pour atteindre la perfection logicielle ? Le travail de **Xavier LEROY** aborde de plain-pied cette question, en mettant un accent particulier sur l'utilisation d'outils de vérification formelle, c'est-à-dire des programmes qui vérifient l'absence de défaillances dans d'autres programmes pouvant contenir des centaines de milliers de lignes de code. Ces outils fournissent des garanties extrêmement précieuses qui complètent et parfois surpassent les résultats obtenus par des techniques plus traditionnelles telles que les tests.

Mais un bug dans l'outil de vérification ou dans le compilateur qui produit l'exécutable à partir de sources vérifiées pourrait remettre en cause ces garanties. Comment pouvons-nous exclure ce risque ? Le **compilateur C vérifié CompCert** développé par **Xavier LEROY** apporte une réponse radicale, fondée sur des bases mathématiques : la vérification formelle, à l'aide d'assistants de preuve, des outils participant à la construction et à la vérification de logiciels critiques.

Langages fonctionnels, systèmes de types et mise en pratique : Caml Light et OCaml

Xavier LEROY a été formé aux mathématiques et à l'informatique à l'École normale supérieure, puis à l'INRIA où il a effectué sa thèse. Programmeur prodige, il commence très tôt dans sa carrière une série de travaux de premier plan sur les **systèmes de types** et les **systèmes de modules** pour les langages fonctionnels. Il se lance en particulier dans le **développement de Caml Light**, devenu aujourd'hui **OCaml**, l'un des deux langages fonctionnels typés les plus utilisés au monde, tant dans la recherche que l'industrie, dans des domaines aussi divers que l'aéronautique, la finance ou le Web. Ce langage, qui est l'une des productions phares d'Inria, est le support de développement d'efforts logiciels rigoureux comme l'**assistant de preuve Coq**, les **analyseurs statiques Astrée et Frama-C**, le **compilateur SCADE 6** d'Estérel Technologies et la ***blockchain* Tezos**. **OCaml** est utilisé dans de nombreux projets emblématiques comme la version web de **Facebook Messenger**, le logiciel **MediaWiki** ou encore l'**infrastructure de virtualisation Docker**.

Preuve de programme, preuve de compilateurs et mise en pratique : le compilateur CompCert

La deuxième grande réalisation de **Xavier LEROY** est celle de **CompCert**, travail dont beaucoup d'experts pensaient qu'il procédait d'un défi impossible à relever. Ce compilateur C certifié, qui a été écrit et vérifié grâce à l'assistant de preuve Coq est une première mondiale à plusieurs titres : il autorise une vérification formelle d'une taille et d'une complexité sans précédents, et surtout, il offre la possibilité de disposer d'un compilateur certifié, étape clé dans la certification et la vérification automatique des chaînes logicielles, et donc vers la programmation « zéro défaut ». Ce fait d'arme a eu un grand retentissement et un impact considérable sur la nature même des grands programmes de recherche sur les logiciels, aussi bien du point de vue de la recherche publique qu'industrielle.

> *« Le développement de logiciels de recherche est le principal dispositif expérimental en informatique : c'est la paillasse du chercheur informaticien. Certains de ces logiciels de recherche mûrissent suffisamment pour sortir du laboratoire et diffuser des idées nouvelles auprès des industriels et du grand public. »*
>
> Xavier LEROY

Les cours de **Xavier LEROY** au Collège de France permettront d'introduire les outils mathématiques et informatiques fondamentaux nécessaires pour comprendre les grandes questions et défis posés par les sciences du logiciel. Son cycle de cours pour l'année académique 2018-2019, ***Programmer = démontrer ? La correspondance de Curry-Howard aujourd'hui***, sera dispensé le mercredi à 10h00 à partir du 21 novembre 2018. Sa leçon inaugurale se déroulera le 15 novembre à 18h. Elle sera ouverte au public et retransmise en direct. L'ensemble de son enseignement sera diffusé sous forme de vidéos sur le portail des savoirs du Collège de France : www.college-de-france.fr.

Les sciences du logiciel

Par Xavier Leroy

Le logiciel, entre l'esprit et la matière

C'est une évidence, l'informatique est aujourd'hui partout : au bureau, à la maison, dans nos téléphones ; au cœur des réseaux de communication, de distribution, d'échanges financiers ; mais aussi, de manière plus discrète car enfouie dans les objets du quotidien, dans les véhicules, les cartes de paiement, les équipements médicaux, les appareils photo, les téléviseurs, l'électroménager, et jusqu'aux ampoules électriques, qui elles aussi deviennent « connectées » et « intelligentes ».

Cette explosion de l'informatique doit beaucoup aux immenses progrès de la micro-électronique, qui débouche sur la production de masse d'ordinateurs et de systèmes sur puce toujours plus puissants à coût constant, mais aussi à l'**incroyable flexibilité du logiciel** qui s'exécute sur ces systèmes. Reprogrammable à l'infini, dupliquable à coût zéro, libéré de presque toute contrainte physique, le logiciel peut atteindre une complexité invraisemblable. Un navigateur Web représente environ 10 millions de lignes de code ; le logiciel embarqué dans une voiture moderne, 100 millions ; l'ensemble des services Internet de Google, 2 milliards. Qu'un assemblage de 2 milliards de choses toutes différentes fonctionne à peu près est sans précédent dans l'histoire de la technologie.

Revers de cette flexibilité, le logiciel est aussi extrêmement **vulnérable aux erreurs de programmation** : les bugs tant redoutés et leur cortège de comportements étranges, de plantages occasionnels, et de mises à jour incessantes. Les failles de sécurité produisent des dégâts considérables, de la demande de rançon à la fuite massive d'informations personnelles. Les libertés fondamentales sont menacées par la généralisation de la surveillance informatique et la manipulation d'élections.

Sur ces problématiques, **les sciences du logiciel apportent un regard objectif, rigoureux, et fondé autant que possible sur les mathématiques**. Quels algorithmes utiliser ? Comment garantir qu'ils sont corrects et efficaces ? Comment les combiner sous forme de programmes ? Dans quels langages exprimer le programme ? Comment compiler ou interpréter ces langages pour les rendre exécutables par la machine ? Se prémunir contre les erreurs de programmation ? Vérifier que le programme final est conforme aux attentes ou n'a pas de comportements indésirables menant à une faille de sécurité ? Voilà des questions de base, qui se posent de tout développement logiciel, et auxquelles les sciences du logiciel apportent des éléments de réponse mathématiquement rigoureux, de l'analyse d'algorithmes à la sémantique des langages de programmation, des spécifications formelles à la vérification des programmes.

À la recherche du logiciel parfait

Les sciences du logiciel nous enseignent qu'un fois donnée une sémantique formelle aux langages de programmation utilisés, **tout programme est aussi une définition mathématique** : on peut raisonner sur le comportement du programme en démontrant des théorèmes sur cette définition ; on peut aussi synthétiser un programme « correct par construction » à partir des propriétés qu'il doit vérifier. Voilà qui peut surprendre tant ces approches, connues sous le nom de « méthodes formelles », diffèrent des

méthodes expérimentales utilisées par tous : on écrit d'abord le programme puis on le teste longuement pour le valider.
Bien qu'efficace pour le logiciel ordinaire, le test devient très coûteux aux plus hauts niveaux d'exigence de qualité, et ne parvient pas à montrer l'absence totale de bugs. **Les méthodes formelles ne souffrent pas de ces limitations et dessinent un chemin vers le logiciel zéro défaut**. Cependant, elles sont longtemps restées à l'état de curiosités académiques, tant les objets mathématiques correspondant à un programme réaliste sont énormes et difficiles à manier. Il a fallu développer des outils de vérification formelle qui automatisent une grande partie des raisonnements pour permettre les premières utilisations systématiques de méthodes formelles pour des logiciels critiques, dans le ferroviaire, l'avionique, et le nucléaire. C'est un des résultats majeurs de la science informatique des vingt dernières années. Ma contribution à cet édifice est d'**étendre la vérification formelle du logiciel critique lui-même aux outils informatiques** (compilateurs, générateurs de code, analyseurs statiques) **qui participent à sa production et sa vérification.**
Après ces premiers succès de la vérification formelle, beaucoup reste à faire pour étendre ses utilisations. Il faut mieux prendre en compte les impératifs de sécurité, le parallélisme (architectures multicœurs et les *GPU* – processeurs graphiques), la distribution (applications Web, *blockchains*). Au centre de l'intelligence artificielle d'aujourd'hui, les techniques d'apprentissage statistiques produisent des logiciels qui ne sont plus complètement écrits mais en grande partie appris à partir d'exemples, nécessitant de nouvelles approches de vérification formelle. Beaucoup de travaux de recherche en perspective, mais toujours l'espoir, grâce à la rigueur des mathématiques et à la puissance des outils informatiques, de maîtriser entièrement le logiciel dans toute sa complexité.

Xavier Leroy

Biographie

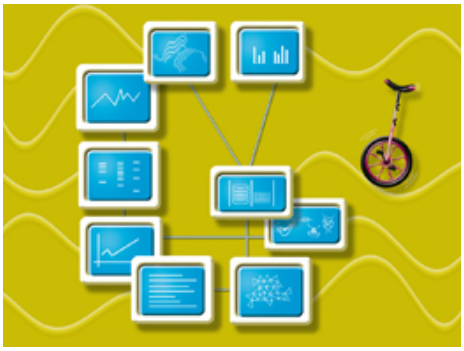
Xavier Leroy a étudié les mathématiques puis l'informatique à l'École Normale Supérieure et à l'Université Paris Diderot. Après un doctorat en informatique fondamentale en 1992 et un post-doctorat à l'Université Stanford, il devient chargé de recherche à l'Inria en 1994, puis directeur de recherche en 2000. Il y a dirige aujourd'hui l'équipe de recherche GALLIUM. De 1999 à 2004, il participe à la start-up Trusted Logic. Il est nommé Professeur au Collège de France en mai 2018 et devient le titulaire de la chaire Sciences du logiciel.

Les travaux de recherche de **Xavier Leroy** portent d'une part sur les nouveaux langages et outils de programmation, et d'autre part sur la vérification formelle de logiciels critiques afin de garantir leur sûreté et leur sécurité. Il est l'architecte et l'un des principaux développeurs du langage de programmation fonctionnelle **OCaml** et du compilateur C formellement vérifié **CompCert**, deux grands logiciels issus de la recherche.

Xavier Leroy a reçu le **prix Michel Monpetit** de l'Académie des Sciences (2007), le **prix Milner** de la Royal Society (2016), le **prix Van Wijngaarden** (2016) du CWI (*Centrum Wiskunde & Informatica*) à Amsterdam, et le **Grand prix Inria-Académie des sciences** (2018). Il est *Fellow* de l'ACM (*Association for Computing Machinery*).

Biographie complète, prix et distinctions : https://www.college-de-france.fr/site/xavier-leroy

Recherche et activités scientifiques :



Les travaux de recherche de l'équipe-projet GALLIUM (Inria) portent sur la conception, la formalisation et l'implémentation de langages et systèmes de programmation. Son objectif est d'améliorer la fiabilité (sûreté de fonctionnement et sécurité) des logiciels en utilisant :

- des langages de programmation de plus haut niveau, plus sûrs et plus expressifs, basés sur le paradigme de la programmation fonctionnelle ;
- la détection automatique d'erreurs de programmation à l'aide de systèmes de types et autres analyses statiques ;
- une meilleure intégration de la programmation et des méthodes formelles, en particulier la preuve sur machine de programmes.

Lien vers les pages de l'équipe-projet GALLIUM : https://www.inria.fr/equipes/gallium

Enseignement du Professeur Xavier LEROY au Collège de France

Cours 2018-2019 : Programmer = démontrer ? La correspondance de Curry-Howard aujourd'hui

Les cours auront lieu le mercredi à 10h00, à partir du 21 novembre 2018.

Informatique et logique mathématique sont historiquement liées : **Alan Turing**, **John von Neumann**, **Alonzo Church** et bien d'autres fondateurs de l'informatique étaient logiciens, professionnels ou de formation. Le cours 2018-2019 de la chaire de Sciences du logiciel étudie un autre lien, de nature mathématique celui-là (il s'agit d'un **isomorphisme**), entre **langages de programmation** et **logiques mathématiques**. Dans cette approche, démontrer un théorème, c'est la même chose que d'écrire un programme ; énoncer le théorème, c'est la même chose que de spécifier partiellement un programme en donnant le type qu'il doit avoir.
Cette **correspondance entre démonstration et programmation** a d'abord été observée dans un cas très simple par deux logiciens : Haskell Curry en 1958 puis William Howard en 1969. Le résultat semblait tellement anecdotique qu'Howard ne l'a jamais soumis à une revue, se contentant de faire circuler des photocopies de ses notes manuscrites. Rarement photocopie a eu un tel impact scientifique, tant cette correspondance de Curry-Howard est entrée en résonance avec le renouveau de la logique et l'explosion de l'informatique théorique des années 1970 pour s'imposer dès 1980 comme un lien structurel profond entre langages et logiques, entre programmation et démonstration. Aujourd'hui, il est naturel de se demander quelle est la signification « logique » de tel ou tel trait de langages de programmation, ou encore quel est le « contenu calculatoire » de tel ou tel théorème mathématique (c'est-à-dire, quels algorithmes se cachent dans ses démonstrations ?). Plus important encore, **la correspondance de Curry-Howard a débouché sur des outils informatiques comme Coq et Agda** qui sont en même temps des langages de programmation et des logiques mathématiques, et s'utilisent aussi bien pour écrire et vérifier des programmes que pour énoncer et aider à démontrer des théories mathématiques.
Le cours retracera ce bouillonnement d'idées à la **frontière entre logique et informatique**, et mettra l'accent sur les résultats récents et les problèmes ouverts dans ce domaine. Pour cette première année, le séminaire donnera la parole à sept experts de ce champ de recherche qui livreront des approfondissements et des points de vue complémentaires.

Les enseignements de **Xavier Leroy** sont ouverts à tous, accessibles gratuitement et sans inscription, dans la limite des places disponibles. Ils seront également diffusés sous forme de vidéos sur le portail des savoirs du Collège de France : www.college-de-france.fr.