Xavier Leroy

Curriculum Vitae

19 mai 2023

Adresse: Collège de France
3, rue d'Ulm
75231 Paris Cedex 05
Mail: xavier.leroy@college-de-france.fr
Web: https://xavierleroy.org/

Résumé

Je suis professeur au Collège de France, titulaire de la chaire de Sciences du logiciel, membre de l'Académie des sciences, et membre de l'équipe Cambium au centre de recherche Inria de Paris. Mes travaux de recherche portent sur les langages et outils de programmation, ainsi que sur la vérification formelle de logiciels par preuve de programme et par analyse statique. Je suis l'architecte et l'un des principaux auteurs du langage de programmation fonctionnelle OCaml et du compilateur C formellement vérifié CompCert.

État-civil

Né le 15 mars 1968. Nationalité française.

Formation

1992	Doctorat d'informatique	Université Paris 7
1989	DEA d'informatique fondamentale	Université Paris 7
1987–1991	Élève	École Normale Supérieure, Paris

Poste actuel

2018–	Professeur au Collège de France, chaire Sciences du logiciel
2022–	Membre de l'Académie des sciences de l'Institut de France

Expérience professionnelle antérieure

2006–2019	Responsable de l'équipe-projet Inria Gallium
2000–2018	Directeur de recherche, Inria
2006–2017	Enseignant au Master MPRI (U. Paris Diderot, ENS Paris, ENS Cachan, Polytechnique)
2012–2016	Délégué scientifique adjoint, centre Inria de Paris
1999–2004	Ingénieur puis conseiller scientifique (à 20%), start-up Trusted Logic
1995–2001	Enseignant au DEA Programmation (U. Paris 6, 7, Sud, ENS Paris)
1994–1999	Chargé de recherche, Inria
1993–1994	Post-doctorant, Stanford University

Domaine de recherche

Mes travaux portent sur les sciences du logiciel, au sens large : comment, à l'aide d'approches rigoureuses et mathématiquement fondées, améliorer la sûreté, la sécurité, et la facilité de développement de logiciels critiques ? Plus précisément, mes principaux thèmes de recherche sont :

- Nouveaux langages de programmation : conception, sémantique formelle, compilation efficace.
- Preuves de programmes et autres méthodes formelles.
- Vérification formelle de compilateurs et autres outils qui participent à la construction du logiciel critique.
- Systèmes de types et autres analyses statiques.
- Systèmes de modules et autres outils pour la programmation à grande échelle.
- Sécurité du logiciel.
- Démonstrations «sur machine», logique pour l'informatique.

La page https://xavierleroy.org/research.html donne plus de détails sur mes travaux.

Prix et distinctions

2023 Prix Lucas de Formal Methods Europe avec S. Blazy and Z. Dargaye.
2022 ACM SIGPLAN Programming Languages Software Award
avec S. Blazy, Z. Dargaye, J. H. Jourdan, M. Schmidt, B. Schommer, J.-B. Tristan
2022 ACM SIGPLAN Programming Languages Achievement Award
2021 ACM Software System Award
avec S. Blazy, Z. Dargaye, J. H. Jourdan, M. Schmidt, B. Schommer, J.-B. Tristan
2018 Grand prix Inria-Académie des sciences
2016 Prix Milner de la Royal Society
2016 Prix Van Wijngaarden du CWI Amsterdam
2015 Fellow of the ACM
2012 Prix Microsoft Research Verified Software Milestone
2011 Prix La Recherche en sciences de l'information, avec S. Blazy, Z. Dargaye et J.-B. Tristan
2007 Prix Michel Monpetit de l'Académie des sciences

Publications

Ouvrages

1. Leroy, X. (déc. 2019b). *Le logiciel, entre l'esprit et la matière*. T. 284. Leçons inaugurales du Collège de France. OpenEdition Books.
2. Weis, P. et X. Leroy (1999). *Le langage Caml*. 2e éd. Dunod. 370 p.
3. Leroy, X. et P. Weis (1993). *Manuel de référence du langage Caml*. InterÉditions. 166 p.

Articles de revues

1. Appel, A. W. et X. Leroy (2023). Efficient Extensional Binary Tries. *Journal of Automated Reasoning* **67**, article 8. DOI: 10.1007/s10817-022-09655-x.
2. Courant, N. et X. Leroy (2021). Verified code generation for the polyhedral model. *Proc. ACM Program. Lang.* **5**(POPL), 40:1-40:24. DOI: 10.1145/3434321.
3. Boldo, S., J.-H. Jourdan, X. Leroy et G. Melquiond (2015). Verified compilation of floating-point computations. *Journal of Automated Reasoning* **54**(2), 135-163. DOI: 10.1007/s10817-014-9317-x.
4. Appel, A. W., R. Dockins et X. Leroy (2012). A list-machine benchmark for mechanized metatheory. *Journal of Automated Reasoning* **49**(3), 453-491. DOI: 10.1007/s10817-011-9226-1.
5. Blazy, S. et X. Leroy (2009). Mechanized semantics for the Clight subset of the C language. *Journal of Automated Reasoning* **43**(3), 263-288. DOI: 10.1007/s10817-009-9148-3.
6. Dargaye, Z. et X. Leroy (2009). A verified framework for higher-order uncurrying optimizations. *Higher-Order and Symbolic Computation* **22**(3), 199-231. DOI: 10.1007/s10990-010-9050-z.
7. Hirschowitz, T., X. Leroy et J. B. Wells (2009). Compilation of extended recursion in call-by-value functional languages. *Higher-Order and Symbolic Computation* **22**(1), 3-66. DOI: 10.1007/s10990-009-9042-z.
8. Leroy, X. (2009a). A formally verified compiler back-end. *Journal of Automated Reasoning* **43**(4), 363-446. DOI: 10.1007/s10817-009-9155-4.
9. Leroy, X. (2009b). Formal verification of a realistic compiler. *Communications of the ACM* **52**(7), 107-115. DOI: 10.1145/1538788.1538814.
10. Leroy, X. et H. Grall (2009). Coinductive big-step operational semantics. *Information and Computation* **207**(2), 284-304. DOI: 10.1016/j.ic.2007.12.004.
11. Leroy, X. et S. Blazy (2008). Formal verification of a C-like memory model and its uses for verifying program transformations. *Journal of Automated Reasoning* **41**(1), 1-31. DOI: 10.1007/s10817-008-9099-0.

12. Rideau, L., B. P. Serpette et X. Leroy (2008). Tilting at windmills with Coq: formal verification of a compilation algorithm for parallel moves. *Journal of Automated Reasoning* **40**(4), 307-326. DOI: 10.1007/s10817-007-9096-8.
13. Hirschowitz, T. et X. Leroy (2005). Mixin modules in a call-by-value setting. *ACM Transactions on Programming Languages and Systems* **27**(5), 857-881. DOI: 10.1145/1086642.1086644.
14. Leroy, X. (2003b). Java bytecode verification: algorithms and formalizations. *Journal of Automated Reasoning* **30**(3–4), 235-269. DOI: 10.1023/A:1025055424017.
15. Leroy, X. (2002). Bytecode verification on Java smart card. *Software – Practice & Experience* **32**(4), 319-340. DOI: 10.1002/spe.438.
16. Leroy, X. (2000). A modular module system. *Journal of Functional Programming* **10**(3), 269-303. DOI: 10.1017/S0956796800003683.
17. Leroy, X. et F. Pessaux (2000). Type-based analysis of uncaught exceptions. *ACM Transactions on Programming Languages and Systems* **22**(2), 340-377. DOI: 10.1145/349214.349230.
18. Hartel, P. H. et al. (1996). Benchmarking implementations of functional languages with "Pseudoknot", a float-intensive benchmark. *Journal of Functional Programming* **6**(4), 621-655.
19. Leroy, X. (1996a). A syntactic theory of type generativity and sharing. *Journal of Functional Programming* **6**(5), 667-698. DOI: 10.1017/S0956796800001933.
20. Leroy, X. et M. Mauny (1993). Dynamics in ML. *Journal of Functional Programming* **3**(4), 431-463. DOI: 10.1017/S0956796800000848.

Chapitres de livres

1. Leroy, X., A. W. Appel, S. Blazy et G. Stewart (mars 2014). "The CompCert memory model". In : *Program Logics for Certified Compilers*. Sous la dir. d'A. W. Appel. Cambridge University Press, pp.237-271.
2. Leroy, X. (2010b). "Mechanized semantics". In : *Logics and languages for reliability and security*. Sous la dir. de J. Esparza, B. Spanfelner et O. Grumberg. NATO Science for Peace and Security Series D: Information and Communication Security 25. IOS Press, pp.195-224. DOI: 10.3233/978-1-60750-100-8-195.
3. Leroy, X. et F. Rouaix (1999). "Security properties of typed applets". In : *Secure Internet Programming – Security issues for Mobile and Distributed Objects*. Sous la dir. de J. Vitek et C. Jensen. LNCS 1603. Springer, pp.147-182. DOI: 10.1007/3-540-48749-2_7.

Articles dans les actes de congrès de premier rang

1. Bourke, T., L. Brun, P.-É. Dagand, X. Leroy, M. Pouzet et L. Rieg (2017). A formally verified compiler for Lustre. *PLDI 2017: Programming Language Design and Implementation*. ACM, pp.586-601. DOI: 10.1145/3062341.3062358.
2. Jourdan, J.-H., V. Laporte, S. Blazy, X. Leroy et D. Pichardie (2015). A formally-verified C static analyzer. *POPL 2015: 42nd symposium Principles of Programming Languages*. ACM, pp.247-259. DOI: 10.1145/2676726.2676966.
3. Boldo, S., J.-H. Jourdan, X. Leroy et G. Melquiond (2013). A formally-verified C compiler supporting floating-point arithmetic. *ARITH 2013: 21st International Symposium on Computer Arithmetic*. IEEE Computer Society, pp.107-115. DOI: 10.1109/ARITH.2013.30.
4. Jourdan, J.-H., F. Pottier et X. Leroy (2012). Validating LR(1) parsers. *ESOP 2012: Programming Languages and Systems, 21st European Symposium on Programming*. LNCS 7211. Springer, pp.397-416. DOI: 10.1007/978-3-642-28869-2_20.
5. Ramananandro, T., G. Dos Reis et X. Leroy (2012). A mechanized semantics for C++ object construction and destruction, with applications to resource management. *POPL 2012: 39th symposium Principles of Programming Languages*. ACM, pp.521-532. DOI: 10.1145/2103656.2103718.
6. Ramananandro, T., G. Dos Reis et X. Leroy (2011). Formal verification of object layout for C++ multiple inheritance. *POPL 2011: 38th symposium Principles of Programming Languages*. ACM, pp.67-79. DOI: 10.1145/1926385.1926395.

7. Tristan, J.-B. et X. Leroy (2010). A simple, verified validator for software pipelining. *POPL 2010: 37th symposium Principles of Programming Languages*. ACM, pp.83-92. DOI: 10.1145/1706299.1706311.
8. Tristan, J.-B. et X. Leroy (2009). Verified validation of Lazy Code Motion. *PLDI 2009: Programming Language Design and Implementation*. ACM, pp.316-326. DOI: 10.1145/1542476.1542512.
9. Tristan, J.-B. et X. Leroy (2008). Formal verification of translation validators: A case study on instruction scheduling optimizations. *POPL 2008: 35th symposium Principles of Programming Languages*. ACM, pp.17-27. DOI: 10.1145/1328897.1328444.
10. Leroy, X. (2006a). Coinductive big-step operational semantics. *ESOP 2006: European Symposium on Programming*. LNCS 3924. Springer, pp.54-68. DOI: 10.1007/11693024_5.
11. Leroy, X. (2006b). Formal certification of a compiler back-end, or: programming a compiler with a proof assistant. *POPL 2006: 33rd symposium Principles of Programming Languages*. ACM, pp.42-54. DOI: 10.1145/1111037.1111042.
12. Hirschowitz, T., X. Leroy et J. B. Wells (2004). Call-by-value mixin modules: reduction semantics, side effects, types. *ESOP 2004: European Symposium on Programming*. LNCS 2986. Springer, pp.64-78. DOI: 10.1007/978-3-540-24725-8_6.
13. Grégoire, B. et X. Leroy (2002). A compiled implementation of strong reduction. *ICFP 2002: International Conference on Functional Programming*. ACM, pp.235-246. DOI: 10.1145/581478.581501.
14. Hirschowitz, T. et X. Leroy (2002). Mixin modules in a call-by-value setting. *ESOP 2002: European Symposium on Programming*. LNCS 2305. Springer, pp.6-20.
15. Leroy, X. (2001a). Java bytecode verification: an overview. *CAV 2001: Computer Aided Verification*. LNCS 2102. Springer, pp.265-285. DOI: 10.1007/3-540-44585-4_26.
16. Pessaux, F. et X. Leroy (1999). Type-based analysis of uncaught exceptions. *POPL 1999: 26th symposium Principles of Programming Languages*. ACM, pp.276-290. DOI: 10.1145/292540.292565.
17. Leroy, X. et F. Rouaix (1998). Security properties of typed applets. *POPL 1998: 25th symposium Principles of Programming Languages*. ACM, pp.391-403. DOI: 10.1145/268946.268979.
18. Leroy, X. (1995a). Applicative functors and fully transparent higher-order modules. *POPL 1995: 22nd symposium Principles of Programming Languages*. ACM, pp.142-153. DOI: 10.1145/199448.199476.
19. Leroy, X. (1994). Manifest types, modules, and separate compilation. *POPL 1994: 21st symposium Principles of Programming Languages*. ACM, pp.109-122. DOI: 10.1145/174675.176926.
20. Doligez, D. et X. Leroy (1993). A concurrent, generational garbage collector for a multithreaded implementation of ML. *POPL 1993: 20th symposium Principles of Programming Languages*. ACM, pp.113-123. DOI: 10.1145/158511.158611.
21. Leroy, X. (1993). Polymorphism by name for references and continuations. *POPL 1993: 20th symposium Principles of Programming Languages*. ACM, pp.220-231. DOI: 10.1145/158511.158632.
22. Leroy, X. (1992d). Unboxed objects and polymorphic typing. *POPL 1992: 19th symposium Principles of Programming Languages*. ACM, pp.177-188. DOI: 10.1145/143165.143205.
23. Leroy, X. et P. Weis (1991). Polymorphic type inference and assignment. *POPL 1991: 18th symposium Principles of Programming Languages*. ACM, pp.291-302. DOI: 10.1145/99583.99622.

Articles dans les actes d'autres congrès internationaux

1. Kästner, D., U. Wünsche, J. Barrho, M. Schlickling, B. Schommer, M. Schmidt, C. Ferdinand, X. Leroy et S. Blazy (jan. 2018). CompCert: Practical experience on integrating and qualifying a formally verified optimizing compiler. *ERTS 2018: Embedded Real Time Software and Systems*. SEE.
2. Leroy, X., S. Blazy, D. Kästner, B. Schommer, M. Pister et C. Ferdinand (2016). CompCert – A formally verified optimizing compiler. *ERTS 2016: Embedded Real Time Software and Systems*. SEE.
3. Krebbers, R., X. Leroy et F. Wiedijk (2014). Formal C semantics: CompCert and the C standard. *ITP 2014: Interactive Theorem Proving*. LNCS 8558. Springer, pp.543-548. DOI: 10.1007/978-3-319-08970-6_36.

4. Bedin França, R., S. Blazy, D. Favre-Felix, X. Leroy, M. Pantel et J. Souyris (2012). Formally verified optimizing compilation in ACG-based flight control software. *ERTS 2012: Embedded Real Time Software and Systems*.
5. Robert, V. et X. Leroy (2012). A formally-verified alias analysis. *CPP 2012: Certified Programs and Proofs*. LNCS 7679. Springer, pp.11-26. DOI: 10.1007/978-3-642-35308-6_5.
6. Rideau, S. et X. Leroy (2010). Validating register allocation and spilling. *CC 2010: Compiler Construction*. LNCS 6011. Springer, pp.224-243. DOI: 10.1007/978-3-642-11970-5_13.
7. Dargaye, Z. et X. Leroy (2007). Mechanized verification of CPS transformations. *LPAR 2007: Logic for Programming, Artificial Intelligence and Reasoning*. LNAI 4790. Springer, pp.211-225. DOI: 10.1007/978-3-540-75560-9_17.
8. Bertot, Y., B. Grégoire et X. Leroy (2006). A structured approach to proving compiler optimizations based on dataflow analysis. *TYPES 2004: Types for Proofs and Programs*. LNCS 3839. Springer, pp.66-81. DOI: 10.1007/11617990_5.
9. Blazy, S., Z. Dargaye et X. Leroy (2006). Formal verification of a C compiler front-end. *FM 2006: Int. Symp. on Formal Methods*. LNCS 4085. Springer, pp.460-475. DOI: 10.1007/11813040_31.
10. Di Cosmo, R., B. Durak, X. Leroy, F. Mancinelli et J. Vouillon (2006). Maintaining large software distributions: new challenges from the FOSS era. *FRCSS 2006: Future Research Challenges for Software and Services*. EASST Newsletter 12, pp.7-20.
11. Mancinelli, F., R. Di Cosmo, J. Vouillon, J. Boender, B. Durak, X. Leroy et R. Treinen (2006). Managing the complexity of large free and open source package-based software distributions. *ASE 2006: 21st Int. Conf. on Automated Software Engineering*. IEEE Computer Society, pp.199-208. DOI: 10.1109/ASE.2006.49.
12. Abiteboul, S., C. Bryce, R. Di Cosmo, K. R. Dittrich, S. Fermigier, S. Laurière, F. Lepied, X. Leroy, T. Milo, E. Panto, R. Pop, A. Sagi, Y. Shtossel, F. Villard et B. Vrdoljak (2005). EDOS: Environment for the Development and Distribution of Open Source Software. *OSS 2005: International Conference on Open Source Systems*.
13. Blazy, S. et X. Leroy (2005). Formal verification of a memory model for C-like imperative languages. *ICFEM 2005: International Conference on Formal Engineering Methods*. LNCS 3785. Springer, pp.280-299. DOI: 10.1007/11576280_20.
14. Calcagno, C., W. Taha, L. Huang et X. Leroy (2003). Implementing multi-stage languages using ASTs, gensym, and reflection. *GPCE 2003: Generative Programming and Component Engineering*. LNCS 2830. Springer, pp.57-76. DOI: 10.1007/978-3-540-39815-8_4.
15. Hirschowitz, T., X. Leroy et J. B. Wells (2003b). Compilation of extended recursion in call-by-value functional languages. *PPDP 2003: International Conference on Principles and Practice of Declarative Programming*. ACM, pp.160-171. DOI: 10.1145/888251.888267.
16. Leroy, X. (2001c). On-card bytecode verification for Java Card. *E-SMART 2001: Smart card programming and security*. LNCS 2140. Springer, pp.150-164. DOI: 10.1007/3-540-45418-7_13.
17. Leroy, X. (mars 1998b). An overview of Types in Compilation. *TIC 1998: workshop Types in Compilation*. LNCS 1473. Springer, pp.1-8. DOI: 10.1007/BFb0055509.
18. Leroy, X. et M. Mauny (1991). Dynamics in ML. *FPCA 1991: Functional Programming Languages and Computer Architecture*. LNCS 523. Springer, pp.406-426.
19. Cardelli, L. et X. Leroy (1990b). Abstract types and the dot notation. *Proceedings IFIP TC2 working conference on programming concepts and methods*. North-Holland, pp.479-504.
20. Leroy, X. (1990b). Efficient data representation in polymorphic languages. *PLILP 1990: Programming Language Implementation and Logic Programming*. LNCS 456. Springer.

Articles dans des workshops ou des congrès nationaux

1. Schommer, B., C. Cullmann, G. Gebhard, X. Leroy, M. Schmidt et S. Wegener (juill. 2018). Embedded Program Annotations for WCET Analysis. *WCET 2018: 18th International Workshop on Worst-Case Execution Time Analysis*. T. 63. OASIcs. Dagstuhl Publishing. DOI: 10.4230/OASIcs.WCET.2018.8.

2. Kästner, D., X. Leroy, S. Blazy, B. Schommer, M. Schmidt et C. Ferdinand (2017). Closing the gap – The formally verified optimizing compiler CompCert. *SSS'17: Developments in System Safety Engineering: Proceedings of the Twenty-fifth Safety-critical Systems Symposium*. CreateSpace, pp.163-180.
3. Bedin França, R., D. Favre-Felix, X. Leroy, M. Pantel et J. Souyris (2011). Towards formally verified optimizing compilation in flight control software. *PPES 2011: Predictability and Performance in Embedded Systems*. OASIcs 18. Dagstuhl Publishing, pp.59-68. DOI: 10.4230/OASIcs.PPES.2011.59.
4. Appel, A. W. et X. Leroy (2007). A list-machine benchmark for mechanized metatheory (extended abstract). *LFMTP 2006: Int. Workshop on Logical Frameworks and Meta-Languages*. ENTCS 174/5, pp.95-108. DOI: 10.1016/j.entcs.2007.01.020.
5. Danelutto, M., R. Di Cosmo, X. Leroy et S. Pelagatti (1998). Parallel functional programming with skeletons: the OCamlP3L experiment. *Proceedings ACM workshop on ML and its applications*. Cornell University.
6. Leroy, X. (juin 1997). The effectiveness of type-based unboxing. *TIC 1997: workshop Types in Compilation*. Technical report BCCS-97-03, Boston College, Computer Science Department.
7. Leroy, X. (jan. 1996b). Le système Caml Special Light: modules et compilation efficace en Caml. *Actes des Journées Francophones des Langages Applicatifs*. INRIA, pp.111-131.
8. Leroy, X. (fév. 1995c). Lessons learned from the translation of documentation from LaTeX to HTML. *ERCIM/W4G Workshop on WWW Authoring and Integration Tools*.
9. Aponte, M.-V. et X. Leroy (1994). Llamado de procedimientos a distancia y abstracción de tipos. *Proc. 20th CLEI PANEL latino-american computer science conference*, pp.1281-1292.

Vulgarisation scientifique

1. Leroy, X. (août 2019a). In search of software perfection: An introduction to deductive software verification. *BOB Summer 2019 Konferenz*. Berlin.
2. Leroy, X. (juill. 2017). How I found a crash bug with hyperthreading in Intel's Skylake processors. *The Next Web*.
3. Leroy, X. (nov. 2016b). In search of software perfection. *Milner award lecture*. London : Royal Society.
4. Leroy, X. (nov. 2016c). On programming languages and their trustworthy implementation. *Van Wijngaarden award ceremony*. Amsterdam : Centrum Wiskunde & Informatica.
5. Leroy, X. (oct. 2015). Desperately seeking software perfection. *Colloquium d'informatique de l'UPMC Sorbonne Universités*. Paris.
6. Leroy, X. (avr. 2014d). Proof assistants in computer science research. *Semantics of proofs and certified mathematics*. Paris : Institut Henri Poincaré.
7. Leroy, X. et J. Souyris (fév. 2014). Développement formel avec Coq. *Preuve de modèle, preuve de programme*. Toulouse : Forum Méthodes Formelles.
8. Leroy, X. (avr. 2010a). Comment faire confiance à un compilateur? *La Recherche* **440**.

Développement de logiciels

CompCert Compilateur vérifié pour logiciels embarqués critiques. CompCert est le premier compilateur réaliste qui a été formellement vérifié à l'aide de méthodes formelles (l'assistant de preuves Coq). Commercialisé par la société AbsInt GmbH et en cours d'évaluation chez Airbus.

OCaml Langage de programmation fonctionnelle statiquement typé et son implémentation. OCaml est, avec Haskell, un des deux langages fonctionnels typés les plus utilisés dans l'industrie comme dans la recherche.

Caml Light Un précurseur du langage OCaml qui l'a fait rentrer dans l'enseignement, notamment dans le cadre de l'option Informatique des classes préparatoires scientifiques.

LinuxThreads Une implémentation de la norme POSIX 1003.1c pour la programmation parallèle. De 1998 à 2004, LinuxThreads a été la bibliothèque de *threads* standard dans les distributions de Linux. Elle a été ensuite remplacée par l'implémentation NPTL.

Composants logiciels pour Java Card (cartes à puces Java) En particulier une machine virtuelle Java Card hautes performances, intégrée dans la carte Odyssey 1 de Bull CP8, et le premier vérificateur embarqué de code JavaCard, commercialisé par Axalto sous le nom Codeshield.

Exposés invités et tutoriels

1. *Formal verification of a code generator for a modeling language : the Velus project*, MARS/VPT workshop, congrès ETAPS, Thessaloniki, avril 2018.
2. *Trust in compilers, code generators, and software verification tools*, ERTS, Toulouse, jan 2018.
3. *In search of software perfection*, exposé du prix Milner, Royal Society, London, nov 2016.
4. *On programming languages and their trustworthy implementation*, remise du prix Van Wijngaarden, Amsterdam, nov 2016.
5. *Formally verifying a compiler : what does it mean, exactly ?*, ICALP 2016, juillet 2016.
6. *Desperately seeking software perfection*, colloquium d'informatique de l'UPMC Sorbonne Universités, oct 2015.
7. *Compiler verification for fun and profit*, FMCAD 2014.
8. *Formal proofs of code generation and verification tools*, SEFM 2014.
9. *Proof assistants in computer science research*, exposé inaugural, trimestre thématique IHP *Semantics of proofs and certified mathematics*, avril 2014.
10. *Mechanized semantics for compiler verification*, APLAS/CPP 2012.
11. *Mechanized verification of program transformations and static analyses*, école d'été Oregon Programming Languages, 2010, 2011, 2012.
12. *Verified squared : does critical software deserve verified tools ?*, POPL 2011.
13. *Mechanized semantics*, école d'été Marktoberdorf 2009.
14. *Du langage à l'action : compilation et typage*, séminaire du cours de Gérard Berry au Collège de France en 2008.
15. *Formal verification of an optimizing compiler*, RTA 2007, MEMOCODE 2007, LCTES 2008.
16. *From Krivine's machine to the Caml implementations*, KAZAM workshop, 2005.
17. *Language-based security for mobile code, with applications to smart cards*, cours à l'école d'hiver TECS Week 2005, Pune, India.
18. *Exploiting type systems and static analyses for smart card security*, CASSIS conference, 2004.
19. *Computer security from a programming language and static analysis perspective*, fédération de congrès ETAPS, avril 2003.
20. *Java bytecode verification : an overview*, Computer Aided Verification, 2001.
21. *Objects and classes vs. modules in Objective Caml*, International Conference on Functional Programming, 1999.
22. *Security properties of typed applets*, workshop APPIA-GULP-PRODE on declarative programming, 1998.
23. *Compilation techniques for functional and object-oriented languages*, Programming Language Design and Implementation, 1998.
24. *An introduction to compiling functional languages*, workshop Types in Compilation, 1998.

Direction de thèses

1. Nathanaëlle Courant (depuis 2019).
2. Basile Clément (2019-2022), co-encadré avec Albert Cohen.

3. Jacques-Henri Jourdan (2012–2016). Jacques-Henri est actuellement chargé de recherche au CNRS.
4. Tahina Ramananandro (2007–2011). Tahina est actuellement ingénieur R&D chez Microsoft (Redmond, USA).
5. Jean-Baptiste Tristan (2006–2009). Jean-Baptiste est actuellement chercheur chez AWS et professeur associé au Boston College.
6. Zaynah Dargaye (2005–2009). Zaynah travaille actuellement chez Nomadic Labs à Paris.
7. Tom Hirschowitz (2000–2003). Tom est actuellement chargé de recherche au CNRS.
8. Benjamin Grégoire (1999–2003), co-encadré avec Benjamin Werner. Benjamin est actuellement chargé de recherche à l'Inria.
9. François Pessaux (1997–1999), co-encadré avec Christian Queinnec. François est actuellement enseignant-chercheur à l'ENSTA ParisTech.

Responsabilités éditoriales

2021–	membre du comité de rédaction de la revue TheoretiCS
2015–	rédacteur du Journal of the ACM, rubrique langages de programmation
2012–2017	membre du comité de rédaction du magazine Communications of the ACM
2007–	membre du comité de rédaction de la revue Journal of Automated Reasoning
2007–2012	co-rédacteur en chef de la revue Journal of Functional Programming.

Autres activités d'évaluation

Président du comité de sélection des congrès POPL 2004 et ICFP 2001, et du workshop TIC 1998. Co-président du comité de sélection du congrès CPP 2015. Participation à environ 40 comités de sélection de congrès internationaux, dont 19 fois pour des congrès de premier rang (POPL, PLDI, ICFP, ESOP).

Membre du comité scientifique du programme SESUR 2007 de l'ANR. Évaluateur AERES/HCERES des laboratoires PPS (2008), VERIMAG (2010), et LORIA (2016).

Autres activités

2012–2016	Membre nommé de la Commission d'Évaluation de l'Inria.
2006–2012	Représentant de l'Inria au comité de direction du master MPRI, et membre de la commission des études de ce master.
2004, 2005	Président de la section locale d'audition pour le recrutement des chargés de recherche de 2e classe à l'Inria Rocquencourt.
1998–2002	Membre élu de la Commission d'Évaluation de l'Inria.
1998–	Membre du groupe de travail IFIP 2.8 *Functional Programming*.