Xavier Leroy

Curriculum Vitae

March 2024

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

Summary

I am professor of software sciences at Collège de France, member of the French Academy of Sciences, and member of the Cambium research team at Inria Paris. My research focuses on programming languages and tools, and on the formal verification of software system using program proof and static analysis. I am the architect and lead developer of the functional programming language OCaml and of the formally-verified C compiler CompCert.

Personal details

Born March 15th 1968. French citizen.

Education

1992	PhD in Computer Science	Université Paris 7
1989	Master's in Fundamental Computer Science	Université Paris 7
1987–1991	Student	École Normale Supérieure, Paris

Current position

2018–	Professor, Collège de France (Paris), chair of Software sciences
2022–	Member of the French Academy of Sciences

Previous positions

2006–2019	Head of Inria Gallium research team
2000–2018	Senior research scientist, Inria
2006–2017	Teaching at MPRI Master's (U. Paris Diderot, ENS Paris, ENS Cachan, Polytechnique)
2012–2016	Deputy scientific vice-president, Inria Paris research center
1999–2004	Engineer, then scientific advisor (20% part-time), Trusted Logic start-up company
1995–2001	Teaching at Master's "Programming" (U. Paris 6, 7, Sud, ENS Paris)
1994–1999	Research scientist, Inria
1993–1994	Post-doc, Stanford University

Research summary

My research studies all scientific aspects of software development. How, with the help of rigorous approaches grounded in mathematics, can we improve safety and security of critical software and facilitate its development and verification? In more details, my main research areas are:

- New programming languages: design, formalization, efficient implementation.
- Program proof and other formal methods.
- Formal verification of compilers and other tools that participate in the development of critical software.
- Type systems and other static analyses.
- Module systems and other linguistic support for programming in the large.
- Software security.
- Mechanized proofs, logic for computer science.

The page https://xavierleroy.org/research.html gives a more detailed overview of my contributions.

Honors and awards

2023	ACM SIGPLAN Programming Languages Software Award for the OCaml functional language and system, with the OCaml development team
2023	Open science award for free software from the French ministry of higher education and research, with the OCaml development team
2023	Lucas Award from Formal Methods Europe with S. Blazy and Z. Dargaye.
2022	ACM SIGPLAN Programming Languages Software Award for the CompCert verified compiler, with 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 for the CompCert verified compiler, with S. Blazy, Z. Dargaye, J. H. Jourdan, M. Schmidt, B. Schommer, J.-B. Tristan
2018	Grand prix Inria-French Academy of Sciences
2016	Milner award, Royal Society
2016	Van Wijngaarden award, CWI Amsterdam
2015	Fellow of the ACM
2012	Microsoft Research Verified Software Milestone
2011	Information science award of the *La Recherche* magazine, with S. Blazy, Z. Dargaye and J.-B. Tristan
2007	Michel Monpetit award, French Academy of sciences

Publications

Books

1. Leroy, X. (Dec. 2019b). *Le logiciel, entre l'esprit et la matière*. Vol. 284. Leçons inaugurales du Collège de France. OpenEdition Books.
2. Weis, P. and X. Leroy (1999). *Le langage Caml*. 2nd ed. Dunod. 370 pp.
3. Leroy, X. and P. Weis (1993). *Manuel de référence du langage Caml*. InterÉditions. 166 pp.

Journal articles

1. Appel, A. W. and X. Leroy (2023). Efficient Extensional Binary Tries. *Journal of Automated Reasoning* **67**, article 8. DOI: 10.1007/s10817-022-09655-x.
2. Courant, N. and 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, and 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, and 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. and 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. and 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, and 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. and 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. and 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, and 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. and 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. and 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. DOI: https://doi.org/10.1017/S0956796800001891.
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. and M. Mauny (1993). Dynamics in ML. *Journal of Functional Programming* **3**(4), 431–463. DOI: 10.1017/S0956796800000848.

Book chapters

1. Leroy, X., A. W. Appel, S. Blazy, and G. Stewart (Mar. 2014). "The CompCert memory model". In: *Program Logics for Certified Compilers*. Ed. by A. W. Appel. Cambridge University Press, pp.237–271.
2. Leroy, X. (2010b). "Mechanized semantics". In: *Logics and languages for reliability and security*. Ed. by J. Esparza, B. Spanfelner, and 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. and F. Rouaix (1999). "Security properties of typed applets". In: *Secure Internet Programming – Security issues for Mobile and Distributed Objects*. Ed. by J. Vitek and C. Jensen. LNCS 1603. Springer, pp.147–182. DOI: 10.1007/3-540-48749-2_7.

Papers in proceedings of leading conferences

1. Bourke, T., L. Brun, P.-É. Dagand, X. Leroy, M. Pouzet, and 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, and 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, and 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, and 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, and 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, and 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. and 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. and 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. and 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, and 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. and 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. and 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. and 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. and 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. and 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. and 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.

Papers in proceedings of other international conferences

1. Kästner, D., U. Wünsche, J. Barrho, M. Schlickling, B. Schommer, M. Schmidt, C. Ferdinand, X. Leroy, and 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, and C. Ferdinand (2016). CompCert – A formally verified optimizing compiler. *ERTS 2016: Embedded Real Time Software and Systems*. SEE.
3. Krebbers, R., X. Leroy, and 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, and J. Souyris (2012). Formally verified optimizing compilation in ACG-based flight control software. *ERTS 2012: Embedded Real Time Software and Systems*.
5. Robert, V. and 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. and 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. and 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, and 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, and 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, and 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, and 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, and 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. and 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, and 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, and 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. (Mar. 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. and M. Mauny (1991). Dynamics in ML. *FPCA 1991: Functional Programming Languages and Computer Architecture*. LNCS 523. Springer, pp.406–426.
19. Cardelli, L. and 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.

Papers in workshops or national conferences

1. Leroy, X. (Jan. 2024). Well-founded recursion done right. *CoqPL 2024: The Tenth International Workshop on Coq for Programming Languages*. ACM. London, United Kingdom.
2. Schommer, B., C. Cullmann, G. Gebhard, X. Leroy, M. Schmidt, and S. Wegener (July 2018). Embedded Program Annotations for WCET Analysis. *WCET 2018: 18th International Workshop on Worst-Case Execution Time Analysis*. Vol. 63. OASIcs. Dagstuhl Publishing. DOI: `10.4230/OASIcs.WCET.2018.8`.
3. Kästner, D., X. Leroy, S. Blazy, B. Schommer, M. Schmidt, and 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.
4. Bedin França, R., D. Favre-Felix, X. Leroy, M. Pantel, and 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`.
5. Appel, A. W. and 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`.
6. Danelutto, M., R. Di Cosmo, X. Leroy, and S. Pelagatti (1998). Parallel functional programming with skeletons: the OCamlP3L experiment. *Proceedings ACM workshop on ML and its applications*. Cornell University.
7. Leroy, X. (June 1997). The effectiveness of type-based unboxing. *TIC 1997: workshop Types in Compilation*. Technical report BCCS-97-03, Boston College, Computer Science Department.
8. 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.
9. Leroy, X. (Feb. 1995c). Lessons learned from the translation of documentation from LaTeX to HTML. *ERCIM/W4G Workshop on WWW Authoring and Integration Tools*.
10. Aponte, M.-V. and X. Leroy (1994). Llamado de procedimientos a distancia y abstracción de tipos. *Proc. 20th CLEI PANEL latino-american computer science conference*, pp.1281–1292.

Popular science

1. Leroy, X. (Aug. 2019a). In search of software perfection: An introduction to deductive software verification. *BOB Summer 2019 Konferenz*. Berlin.
2. Leroy, X. (July 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. (Apr. 2014d). Proof assistants in computer science research. *Semantics of proofs and certified mathematics*. Paris: Institut Henri Poincaré.
7. Leroy, X. and J. Souyris (Feb. 2014). Développement formel avec Coq. *Preuve de modèle, preuve de programme*. Toulouse: Forum Méthodes Formelles.
8. Leroy, X. (Apr. 2010a). Comment faire confiance à un compilateur? *La Recherche* **440**.

Edition of proceedings

1. Charpin, D. and X. Leroy, eds. (Sept. 2023). *Déchiffrement(s): des hiéroglyphes à l'ADN*. Colloque annuel du Collège de France. Odile Jacob.
2. Leroy, X. and A. Tiu, eds. (2015). *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*. ACM.
3. Benton, N. and X. Leroy, eds. (2006). *ML 2005: Proceedings of the ACM SIGPLAN Workshop on ML*. ENTCS 148(2). DOI: 10.1016/j.entcs.2005.11.037.
4. Jones, N. D. and X. Leroy, eds. (2004). *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004*. ACM.
5. Pierce, B. C. and X. Leroy, eds. (2001). *Proceedings of the Sixth ACM SIGPLAN International Conference on Functional Programming, ICFP '01, Firenze (Florence), Italy, September 3-5, 2001*. ACM.
6. Leroy, X. and A. Ohori, eds. (1998). *Types in Compilation, Second International Workshop, TIC '98, Kyoto, Japan, March 25-27, 1998, Proceedings*. LNCS 1473. Springer.

Software development

CompCert Formally-verified C compiler for critical embedded software. CompCert is the first realistic compiler that was verified to be free of miscompilation errors using formal methods (the Coq proof assistant). Marketed by the AbsInt GmbH company; under evaluation at Airbus.

OCaml A functional programming language and its implementation. OCaml and Haskell are the two typed functional languages most widely used in industry as well as in academia.

Caml Light An earlier, much simpler version of the OCaml language. It has been widely used for teaching, in particular in French preparatory classes.

LinuxThreads An implementation of the POSIX 1003.1c standard for shared-memory parallel programming. Between 1998 and 2004, LinuxThreads was the standard thread library in Linux distributions. It was then replaced by NPTL.

Java Card software components (Java on smart cards) In particular, a high-performance Java Card virtual machine that was integrated in Bull CP8's Odyssey 1 card, and the first embedded Java Card bytecode verifier, marketed by Axalto under the Codeshield brand.

Invited talks and tutorials

1. *Mechanizing abstract interpretation*, Next 40 Years of Abstract Interpretation workshop, London, 2024.
2. *25 years of OCaml*, OCaml Workshop, online, 2021.
3. *Formal verification of a code generator for a modeling language: the Velus project*, MARS/VPT workshop, ETAPS joint conferences, Thessaloniki, 2018.
4. *Trust in compilers, code generators, and software verification tools*, ERTS, Toulouse, 2018.
5. *In search of software perfection*, Milner award lecture, Royal Society, London, 2016.
6. *On programming languages and their trustworthy implementation*, Van Wijngaarden award ceremony, Amsterdam, 2016.
7. *Formally verifying a compiler: what does it mean, exactly?*, ICALP 2016, 2016.
8. *Desperately seeking software perfection*, colloquium d'informatique de l'UPMC Sorbonne Universités, Paris, 2015.
9. *Compiler verification for fun and profit*, FMCAD, Lausanne, 2014.
10. *Formal proofs of code generation and verification tools*, SEFM, Grenoble, 2014.
11. *Proof assistants in computer science research*, inaugural lecture, thematic quarter IHP *Semantics of proofs and certified mathematics*, Paris, 2014.

12. *Mechanized semantics for compiler verification*, APLAS/CPP, Kyoto, 2012.
13. *Mechanized verification of program transformations and static analyses*, Oregon Programming Languages Summer School, Eugene, 2010, 2011, 2012.
14. *Verified compilers, verified static analyzers, and certified decision procedures*, CADE, Wroclaw, 2011.
15. *Verified squared: does critical software deserve verified tools?*, POPL, Austin, 2011.
16. *Mechanized semantics*, Marktoberdorf 2009 summer school.
17. *Du langage à l'action: compilation et typage*, seminar at Gérard Berry's 2008 course, Collège de France.
18. *Formal verification of an optimizing compiler*, RTA 2007, MEMOCODE 2007, LCTES 2008.
19. *From Krivine's machine to the Caml implementations*, KAZAM workshop, 2005.
20. *Language-based security for mobile code, with applications to smart cards*, lecture at the TECS Week 2005 winter school, Pune, India.
21. *Exploiting type systems and static analyses for smart card security*, CASSIS conference, 2004.
22. *Computer security from a programming language and static analysis perspective*, ETAPS 2003 conferences, unifying speaker lecture.
23. *Java bytecode verification: an overview*, Computer Aided Verification, 2001.
24. *Objects and classes vs. modules in Objective Caml*, International Conference on Functional Programming, 1999.
25. *Security properties of typed applets*, workshop APPIA-GULP-PRODE on declarative programming, 1998.
26. *Compilation techniques for functional and object-oriented languages*, Programming Language Design and Implementation, 1998.
27. *An introduction to compiling functional languages*, workshop Types in Compilation, 1998.

PhD supervision

1. Nathanaëlle Courant (since 2019).
2. Basile Clément (2019-2022), co-supervised with Albert Cohen. Clément is currently with OCamlPro in Paris.
3. Jacques-Henri Jourdan (2012–2016). Jacques-Henri is currently a CNRS research scientist.
4. Tahina Ramananandro (2007–2011). Tahina is currently a R&D engineer at Microsoft (Redmond, USA).
5. Jean-Baptiste Tristan (2006–2009). Jean-Baptiste is currently principal scientist at AWS and associate professor at Boston College.
6. Zaynah Dargaye (2005–2009). Zaynah is currently with the Nomadic Labs company in Paris.
7. Tom Hirschowitz (2000–2003). Tom is currently a CNRS research scientist.
8. Benjamin Grégoire (1999–2003), co-supervised with Benjamin Werner. Benjamin is currently an Inria research scientist.
9. François Pessaux (1997–1999), co-supervised with Christian Queinnec. François is currently assistant professor at ENSTA ParisTech.

Editorial duties

2021–	TheoretiCS, associate editor
2015–2023	Journal of the ACM, area editor for programming languages
2012–2017	Communications of the ACM, member of the editorial board, Research Highlights section
2007–	Journal of Automated Reasoning, associate editor
2007–2012	Journal of Functional Programming, co-editor in chief.

Other evaluation activities

Program chair for the POPL 2004 and ICFP 2001 conferences and for the TIC 1998 workshop. Co-program chair for the CPP 2015 conference. Member of about 40 program committees of international conferences, including 20 committees for leading conferences (POPL, PLDI, ICFP, ESOP).

Member of the scientific committee for the SESUR 2007 funding program of ANR. Member of the AERES/HCERES evaluation committees for the French laboratories PPS (2008), VERIMAG (2010), and LORIA (2016).

Other duties

2012–2016	Appointed member of Inria's Evaluation Committee.
2006–2012	Inria representative on the board of directors of MPRI master's, and member of its studies committee..
2004, 2005	Chair of the hiring committee for junior research scientists at Inria Rocquencourt.
1998–2002	Elected member of Inria's Evaluation Committee.
1998–	Member of IFIP Working Group 2.8 *Functional Programming*.