Static Analysis and Verification of Aerospace Software by Abstract Interpretation

---

Julien Bertrane
Département d'informatique, École normale supérieure

Patrick Cousot
Département d'informatique, École normale supérieure &
Courant Institute of Mathematical Sciences, New York University

Radhia Cousot
CNRS & Département d'informatique, École normale supérieure

Jérôme Feret
INRIA & Département d'informatique, École normale supérieure

Laurent Mauborgne
AbsInt Angewandte Informatik

Antoine Miné
Sorbonne University, University Pierre and Marie Curie, CNRS, LIP6

Xavier Rival
INRIA & Département d'informatique, École normale supérieure

now
the essence of knowledge
Boston — Delft

Full text available at: http://dx.doi.org/10.1561/2500000002

Foundations and Trends® in Programming Languages

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

J. Bertrane *et al.*. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. *Foundations and Trends® in Programming Languages, vol. 2, no. 2-3, pp. 71–190, 2015.*

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-60198-857-7
© 2015 J. Bertrane *et al.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Foundations and Trends® in Programming Languages

Volume 2, Issue 2-3, 2015

Editorial Board

Editor-in-Chief

Editors

Editorial Scope

Topics

Foundations and Trends® in Programming Languages publishes survey and tutorial articles in the following topics:

- Abstract interpretation
- Compilation and interpretation techniques
- Domain specific languages
- Formal semantics, including lambda calculi, process calculi, and process algebra
- Language paradigms
- Mechanical proof checking
- Memory management
- Partial evaluation
- Program logic
- Programming language implementation
- Programming language security
- Programming languages for concurrency
- Programming languages for parallelism
- Program synthesis
- Program transformations and optimizations
- Program verification
- Runtime techniques for programming languages
- Software model checking
- Static and dynamic program analysis
- Type theory and type systems

Information for Librarians

Foundations and Trends® in Programming Languages, 2015, Volume 2, 4 issues. ISSN paper version 2325-1107. ISSN online version 2325-1131. Also available as a combined paper and online subscription.

Foundations and Trends® in Programming Languages
Vol. 2, No. 2-3 (2015) 71–190

DOI: 10.1561/2500000002

Static Analysis and Verification of Aerospace Software by Abstract Interpretation

Julien Bertrane
Département d'informatique, École normale supérieure

Patrick Cousot
Département d'informatique, École normale supérieure &
Courant Institute of Mathematical Sciences, New York University

Radhia Cousot
CNRS & Département d'informatique, École normale supérieure

Jérôme Feret
INRIA & Département d'informatique, École normale supérieure

Laurent Mauborgne
AbsInt Angewandte Informatik

Antoine Miné
Sorbonne University, University Pierre and Marie Curie, CNRS, LIP6

Xavier Rival
INRIA & Département d'informatique, École normale supérieure

Contents

Full text available at: http://dx.doi.org/10.1561/2500000002

Abstract

We discuss the principles of static analysis by abstract interpretation and report on the automatic verification of the absence of runtime errors in large embedded aerospace software by static analysis based on abstract interpretation. The first industrial applications concerned synchronous control/command software in open loop. Recent advances consider imperfectly synchronous programs, parallel programs, and target code validation as well. Future research directions on abstract interpretation are also discussed in the context of aerospace software.

J. Bertrane, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. *Static Analysis and Verification of Aerospace Software by Abstract Interpretation*. Foundations and Trends® in Programming Languages, vol. 2, no. 2-3, pp. 71–190, 2015.

DOI: 10.1561/2500000002.

Nomenclature

1_S	identity on S (also t^0)
$t \circ r$	composition of relations t and r
t^n	powers of relation t
$t^\star$	reflexive transitive closure of relation t
$\mathrm{lfp}^{\subseteq} F$	least fixpoint of F for $\subseteq$
$\wp(S)$	parts of set S (also 2^S)
$\mathtt{x}$	program variable
$\mathtt{V}$	set of all program variables
S	program states
I	initial states
t	state transition relation
$\mathcal{C}[\![t]\!]I$	collecting semantics
$\mathcal{T}[\![t]\!]I$	trace semantics
$\mathcal{R}[\![t]\!]I$	reachability semantics
$\mathcal{P}[\![t]\!]I$	prefix trace semantics
F_P	prefix trace transformer
F_ι	interval transformer
F_R	reachability transformer
α	abstraction function
γ	concretization function
$X^\sharp$	abstract counterpart of X
ρ	reduction
∇	widening
Δ	narrowing
$\mathbb{Z}$	integers
$\mathbb{N}$	naturals
$\mathbb{R}$	reals
$\lvert x \rvert$	absolute value of x
q	quaternion
$\lvert\lvert q \rvert\rvert$	norm of quaternion q
$\bar{q}$	conjugate of quaternion q

1

Introduction

The validation of software checks informally (e.g., by code reviews or tests) the conformance of the software executions to a specification. More rigorously, the verification of software proves formally the conformance of the software semantics (that is, the set of all possible executions in all possible environments) to a specification. It is of course difficult to design a sound semantics, to get a rigorous description of all execution environments, to derive an automatically exploitable specification from informal natural language requirements, and to completely automatize the formal conformance proof (which is undecidable). In model-based design, the software is often generated automatically from the model so that the certification of the software requires the validation or verification of the model plus that of the translation into an executable software (through compiler verification or translation validation). Moreover, the model is often considered to be the specification, so there is no specification of the specification, hence no other possible conformance check. These difficulties show that fully automatic rigorous verification of complex software is very challenging and perfection is impossible.

We present abstract interpretation [Cousot and Cousot, 1977] and show how its principles can be successfully applied to cope with the above-mentioned difficulties inherent to formal verification.

First, semantics and execution environments can be precisely formalized at different levels of abstraction, so as to correspond to a pertinent level of description as required for the formal verification.

Second, semantics and execution environments can be over-approximated, since it is always sound to consider, in the verification process, more executions and environments than actually occurring in real executions of the software. It is crucial for soundness, however, to never omit any of them, even rare events. For example, floating-point operations incur rounding (to nearest, towards 0, plus or minus infinity) and, in the absence of precise knowledge of the execution environment, one must consider the worst case for each floating-point operation. Another example is the range of inputs, like voltages, that can be overestimated by the full range of the hardware register where the value is sampled (anyway, a well-designed software should be defensive, i.e., have appropriate protections to cope with erroneous or failing sensors and be prepared to accept any value from the registers).

In the absence of an explicit formal specification or to avoid the additional cost of translating the specification into a format understandable by the verification tool, one can consider implicit specifications. For example, memory leaks, buffer overruns, undesired modulo in integer arithmetics, floating-point overflows, data-races, deadlocks, live-locks, etc. are all frequent symptoms of software bugs, the absence of which can be easily incorporated as a valid but incomplete specification in a verification tool, maybe using user-defined parameters to choose among several plausible alternatives.

Because of undecidability issues (which make fully automatic proofs on all programs ultimately impossible) and the desire not to rely on end-user interactive help (which can add a heavy, or even intractable cost), abstract interpretation makes an intensive use of the idea of abstraction, either to restrict the properties to be considered (which introduces the possibility to have efficient computer representations and algorithms to manipulate them) or to approximate the solutions of the equations involved in the definition of the abstract semantics. Thus, proofs can be automated in a way that is always sound but may be imprecise, so that some questions about the program behaviors and the conformance to the specification cannot be definitely answered neither affirmatively nor negatively. So, for soundness, an alarm will be raised which may be false. Intensive research work is done to discover appropriate abstractions eliminating this uncertainty about false alarms for domain-specific applications.

In this article, we report on the successful scalable and cost-effective application of abstract interpretation to the verification of the absence of runtime errors in aerospace control software by the ASTRÉE static analyzer [Cousot et al., 2007a], illustrated first by the verification of the fly-by-wire primary software of commercial airplanes [Delmas and Souyris, 2007] and then by the validation of the Monitoring and Safing Unit (MSU) of the Jules Vernes ATV docking software [Bouissou et al., 2009]. We also discuss on-going extensions to imperfectly synchronous software, parallel software, and target code validation, and conclude with more prospective goals for rigorously verifying and validating aerospace software.

An early version of this article appeared in [Bertrane et al., 2010]. We extend that version with more thorough explanations, additional examples, and updated experimental results.

References

AbsInt, Angewandte Informatik. Astrée run-time error analyzer. `http://www.absint.com/astree/`.

Aeronautical Radio, Inc. (ARINC). ARINC 653. `http://www.arinc.com/`.

J. Bertrane. Static analysis by abstract interpretation of the quasi-synchronous composition of synchronous programs. In R. Cousot, editor, *Proc. of the 6th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 97–112. Springer (Berlin), 2005.

J. Bertrane. Proving the properties of communicating imperfectly-clocked synchronous systems. In Kwangkeun Yi, editor, *Proc. of the 13th Int. Static Analysis Symposium (SAS'06)*, volume 4134 of *LNCS*, pages 370–386. Springer (Berlin), 2006.

J. Bertrane, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. Static analysis and verification of aerospace software by abstract interpretation. In *AIAA Infotech@Aerospace (I@A 2010)*, number AIAA-2010-3385, pages 1–38. AIAA (American Institute of Aeronautics and Astronautics), 2010.

F. Besson, D. Cachera, T.P. Jensen, and D. Pichardie. Certified static analysis by abstract interpretation. In *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*, volume 5705 of *LNCS*, pages 223–257. Springer (Berlin), 2009.

B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software, invited chapter. In T. Mogensen, D.A. Schmidt, and I.H. Sudborough, editors, *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, volume 2566 of *LNCS*, pages 85–108. Springer (Berlin), 2002.

B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. of the ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation (PLDI'03)*, pages 196–207. ACM Press (New York), 2003.

O. Bouissou, E. Conquet, P. Cousot, R. Cousot, J. Feret, E. Goubault, K. Ghorbal, D. Lesens, L. Mauborgne, A. Miné, S. Putot, X. Rival, and M. Turin. Space software validation using abstract interpretation. In *Proc. of the Int. Space System Engineering Conference, Data Systems In Aerospace (DASIA'09)*, pages 1–7. ESA publications, 2009.

F. Bourdoncle. Abstract interpretation by dynamic partitioning. *Journal of Functional Programming*, 2(4):407–423, 1992.

F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In *Proc. of the Int. Conf. on Formal Methods in Programming and their Applications (FMPA'93)*, volume 735 of *LNCS*, pages 128–14. Springer (Berlin), 1993.

R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

R. M. Burstall. Program proving as hand simulation with a little induction. In *Proc. of IFIP Congress*, pages 308–312, 1974.

P. Caspi, C. Mazuet, and N. Reynaud Paligot. About the design of distributed control systems: The quasi-synchronous approach. In Udo Voges, editor, *20th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2001)*, volume 2187 of *LNCS*, pages 215–226. Springer (Berlin), 2001.

P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes (in French).* Thèse d'État ès sciences mathématiques, Université scientifique et médicale de Grenoble, Grenoble, France, 21 Mar. 1978.

P. Cousot. Semantic foundations of program analysis, invited chapter. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, Inc., Englewood Cliffs, 1981.

P. Cousot. Types as abstract interpretations. In *Conf. Rec. of the 24th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'97)*, pages 316–331. ACM Press (New York), 1997.

P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science*, 277(1–2):47–103, 2002.

P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. of the Second Int. Symp. on Programming (ISOP'76)*, pages 106–130. Dunod, Paris, 1976.

P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conf. Rec. of the 4th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'77)*, pages 238–252. ACM Press (New York), 1977.

P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 81(1):43–57, 1979a.

P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conf. Rec. of the 6th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'79)*, pages 269–282. ACM Press (New York), 1979b.

P. Cousot and R. Cousot. *Invariance proof methods and analysis techniques for parallel programs*, chapter 12, pages 243–271. Macmillan, 1984.

P. Cousot and R. Cousot. Sometime = always + recursion ≡ always: on the equivalence of the intermittent and invariant assertions methods for proving inevitability properties of programs. *Acta Informatica*, 24:1–31, 1987.

P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, Aug. 1992a.

P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proc. of the 4th Int. Symp. on Programming Language Implementation and Logic Programming (PLILP'92)*, volume 631 of *LNCS*, pages 269–295. Springer (Berlin), 1992b.

P. Cousot and R. Cousot. "À la Burstall" intermittent assertions induction principles for proving inevitability properties of programs. *Theoretical Computer Science*, 120:123–155, 1993.

P. Cousot and R. Cousot. Systematic design of program transformation frameworks by abstract interpretation. In *Conf. Rec. of the 29th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'02)*, pages 178–190. ACM Press (New York), 2002.

P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conf. Rec. of the 5th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'78)*, pages 84–97. ACM Press (New York), 1978.

P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. The ASTRÉE static analyzer. `www.astree.ens.fr` and `www.absint.com/astree/`.

P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE analyser. In M. Sagiv, editor, *Proc. of the 14th European Symposium on Programming Languages and Systems (ESOP'05)*, volume 3444 of *LNCS*, pages 21–30. Springer (Berlin), 2005.

P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Combination of abstractions in the ASTRÉE static analyzer. In M. Okada and I. Satoh, editors, *Proc. of the 11th Annual Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *LNCS*, pages 272–300. Springer (Berlin), 2006.

P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Varieties of static analyzers: A comparison with ASTRÉE. In M. Hinchey, He Jifeng, and J. Sanders, editors, *Proc. of the First Symp. on Theoretical Aspects of Software Engineering (TASE'07)*, pages 3–17. IEEE Computer Society Press, 2007a.

P. Cousot, P. Ganty, and J.-F. Raskin. Fixpoint-guided abstraction refinements. In G. Filé and H. Riis-Nielson, editors, *Proc. of the 14th Int. Static Analysis Symposium (SAS'07)*, volume 4634 of *LNCS*, pages 333–348. Springer (Berlin), 2007b.

P. Cousot, R. Cousot, and L. Mauborgne. The reduced product of abstract domains and the combination of decision procedures. In M. Hofmann, editor, *14th Int. Conf. on Fondations of Software Science and Computation Structures (FoSSaCS 2011)*, volume 6604 of *Lecture Notes in Computer Science*, pages 456–472. Springer-Verlag, 2011.

D. Delmas and J. Souyris. ASTRÉE: from research to industry. In G. Filé and H. Riis-Nielson, editors, *Proc. of the 14th Int. Static Analysis Symposium (SAS'07)*, volume 4634 of *LNCS*, pages 437–451. Springer (Berlin), 2007.

E. W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, *Programming Languages: NATO Advanced Study Institute*, pages 43–112. Academic Press, 1968.

dSpace. TargetLink code generator. `http://www.dspaceinc.com`.

Esterel Technologies. Scade suite™, the standard for the development of safety-critical embedded software in the avionics industry. `http://www.esterel-technologies.com/`.

Euclid of Alexandria. Elementa geometriæ, book xii, proposition 17. fl. 300 BC.

J. Feret. Static analysis of digital filters. In D. Schmidt, editor, *Proc. of the 13th European Symp. on Programming Languages and Systems (ESOP'04)*, volume 2986 of *LNCS*, pages 33–48. Springer (Berlin), 2004.

J. Feret. The arithmetic-geometric progression abstract domain. In R. Cousot, editor, *Proc. of the 6th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 42–58. Springer (Berlin), 2005a.

J. Feret. Numerical abstract domains for digital filters. In *Proc. of the First Int. Workshop on Numerical & Symbolic Abstract Domains (NSAD'05)*, 2005b.

P. Ferrara. Static analysis via abstract interpretation of the happens-before memory model. In *Proc. of the Second Int. Conf. on Tests and Proofs (TAP'08)*, volume 4966 of *LNCS*, pages 116–133. Springer (Berlin), 2008.

P. Ferrara. *Static analysis via abstract interpretation of multithreaded programs*. PhD thesis, École Polytechnique, Palaiseau, France, May 2009.

R. W. Floyd. Assigning meanings to programs. In *Proc. of the American Mathematical Society Symposia on Applied Mathematics*, volume 19, pages 19–32. Springer Netherlands, 1967.

R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *Journal of the Association for Computing Machinery*, 47(2):361–416, 2000.

R. Ginosar. Fourteen ways to fool your synchronizer. In *Proc. of the 9th Int. Symposium on Asynchronous Circuits and Systems (ASYNC'03)*, pages 89–97. IEEE Computer Society, 2003.

J. Gosling, B. Joy, G. Steele, and G. Bracha. *The Java language specification, third edition*. Addison Wesley, 2005.

É. Goubault. Static analyses of floating-point operations. In *Proc. of the 8th Int. Static Analysis Symposium (SAS'01)*, volume 2126 of *LNCS*, pages 234–259. Springer (Berlin), 2001.

P. Granger. Static analysis of arithmetical congruences. *International Journal of Computer Mathematics*, 30(3 & 4):165–190, 1989.

P. Granger. Static analysis of linear congruence equalities among variables of a program. In S. Abramsky and T.S.E. Maibaum, editors, *Proc. of the Int. Joint Conf. on Theory and Practice of Software Development (TAPSOFT'91), Volume 1 (CAAP'91)*, volume 493 of *LNCS*, pages 169–192. Springer (Berlin), 1991.

R. Heckmann and C. Ferdinand. Worst-case execution time prediction by static program analysis. In *Proc. of the 18th Int. Parallel and Distributed Processing Symposium (IPDPS'04)*, pages 26–30. IEEE Computer Society, 2004.

IEEE and The Open Group. Portable operating system interface (POSIX). `http://www.opengroup.org`, `http://standards.ieee.org`.

IEEE Computer Society. IEEE standard for binary floating-point arithmetic. Technical report, ANSI/IEEE Std. 745-1985, 1985.

ISO/IEC JTC1/SC22/WG14 Working Group. C standard. Technical Report 1124, ISO & IEC, 2007.

ISO/IEC JTC1/SC22/WG21 Working Group. Working draft, standard for programming language C++. Technical Report 3035, ISO & IEC, 2010.

B. Jeannet and A. Miné. The `Apron` numerical abstract domain library. `http://apron.cri.ensmp.fr/library/`, 2007.

B. Jeannet and A. Miné. `Apron`: A library of numerical abstract domains for static analysis. In *Proc. of the 21st Int. Conf. on Computer Aided Verification (CAV'09)*, volume 5643 of *LNCS*, pages 661–667. Springer (Berlin), 2009.

C. B. Jones. *Development Methods for Computer Programs including a Notion of Interference*. PhD thesis, Oxford University, Jun. 1981.

J.-H. Jourdan, V. Laporte, S. Blazy, X. Leroy, and D. Pichardie. A formally-verified C static analyzer. In *Conf. Rec. of the 42nd Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL 2015)*, pages 247–259. ACM Press (New York), 2015.

D. Kästner, S. Wilhelm, S. Nenova, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. ASTRÉE: Proving the absence of rutime errors. In *Proc. of Embedded Real-Time Software and Systems (ERTS'10)*, pages 1–5, 2010.

L. Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. on Software Engineering*, 3(2):125–143, Mar. 1977.

L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Transactions on Computers*, 28:690–691, Sept. 1979.

K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In *Proc. of the 18th IEEE Real-Time Systems Symp. (RTSS'97)*, pages 14–24. IEEE CS Press, 1997.

X. Leroy. Formal certification of a compiler back-end or: programming a compiler with a proof assistant. In *Conf. Rec. of the 33rd Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'06)*, pages 42–54. ACM Press (New York), 2006.

J. Manson, W. Pugh, and S. V. Adve. The java memory model. In *Conf. Rec. of the 32nd Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'05)*, pages 378–391. ACM Press (New York), 2005.

M. Martel. Enhancing the implementation of mathematical formulas for fixed-point and floating-point arithmetics. *Formal Methods in System Design*, 35 (3):265–278, Dec. 2009.

L. Mauborgne. ASTRÉE: Verification of absence of run-time error. In P. Jacquart, editor, *Building the Information Society*, chapter 4, pages 385–392. Kluwer Academic Publishers, Dordrecht, 2004.

L. Mauborgne and X. Rival. Trace partitioning in abstract interpretation based static analyzer. In M. Sagiv, editor, *Proc. of the 14th European Symp. on Programming Languages and Systems (ESOP'05)*, volume 3444 of *LNCS*, pages 5–20. Springer (Berlin), 2005.

A. Miné. The octagon abstract domain. In *Proc. of the Analysis, Slicing and Transformation Workshop (AST'01)*, pages 310–319. IEEE Computer Society Press (Los Alamitos), 2001.

A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In D. Schmidt, editor, *Proc. of the 13th European Symp. on Programming Languages and Systems (ESOP'04)*, volume 2986 of *LNCS*, pages 3–17. Springer (Berlin), 2004a.

A. Miné. *Weakly Relational Numerical Abstract Domains.* Thèse de doctorat en informatique, École polytechnique, Palaiseau, France, 6 Dec. 2004b.

A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19:31–100, 2006a.

A. Miné. Field-sensitive value analysis of embedded C programs with union types and pointer arithmetics. In *Proc. of the ACM SIGPLAN-SIGBED Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'06)*, pages 54–63. ACM Press (New York), 2006b.

A. Miné. Static analysis of run-time errors in embedded critical parallel C programs. In *Proc. of the 20th European Symposium on Programming (ESOP'11)*, volume 6602 of *LNCS*, pages 398–418. Springer, 2011.

A. Miné. Relational thread-modular static value analysis by abstract interpretation. In *Proc. of the 15th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'14)*, volume 8318 of *Lecture Notes in Computer Science (LNCS)*, pages 39–58. Springer, 2014.

D. Monniaux. The parallel implementation of the ASTRÉE static analyzer. In *Proc. of the 3rd Asian Symp. on Programming Languages and Systems (APLAS'05)*, volume 3780 of *LNCS*, pages 86–96. Springer (Berlin), 2005.

R. E. Moore. *Interval Analysis.* Prentice Hall, Englewood Cliffs N. J., USA, 1966.

G. C. Necula. Proof-Carrying Code. In *Conf. Rec. of the 24th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Langauges (POPL'97)*, pages 106–119. ACM Press (New York), 1997.

G. C. Necula. Translation Validation for an Optimizing Compiler. In *Proc. of the Conf. on Programming Language Design and Implementation (PLDI'00)*, pages 83–94. ACM Press (New York), 2000.

S. Owicki and D. Gries. An axiomatic proof technique for parallel programs i. *Acta Informatica*, 6(4):319–340, Dec. 1976.

A. Pnueli, O. Shtrichman, and M. Siegel. Translation Validation for Synchronous Languages. In *Proc. of the 25th Int. Coll. on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *LNCS*, pages 235–246. Springer (Berlin), 1998.

J. C. Reynolds. The discoveries of continuations. *Lisp and Sy,bolic Computation*, 6(3–4):233–248, 1993.

X. Rival. Abstract Interpretation-based Certification of Assembly Code. In *Proc. of the 4th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'03)*, volume 2575 of *LNCS*, pages 41–55. Springer (Berlin), 2003.

X. Rival. Symbolic transfer functions-based approaches to certified compilation. In *Conf. Rec. of the 31st Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'04)*, pages 1–13. ACM Press (New York), 2004.

X. Rival and L. Mauborgne. The trace partitioning abstract domain. *ACM Transactions on Programming Languages Systems*, 29(5), 2007.

V. Saraswat, R. Jagadeesan, M. Michael, and C. von Praun. A theory of memory models. In *Proc. of the 12th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPOPP'07)*, pages 161–172. ACM Press (New York), 2007.

A. Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–310, 1955.

Radio Technical Commission on Aviation. DO-178B. Technical report, Software Considerations in Airborne Systems and Equipment Certification, 1999.