IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2013

Project-Team ABSTRACTION

Abstract Interpretation and Static Analysis

IN COLLABORATION WITH: Département d'Informatique de l'Ecole Normale Supérieure

RESEARCH CENTER
Paris - Rocquencourt

THEME
Proofs and Verification

Table of contents

Project-Team ABSTRACTION

Keywords: Abstract Interpretation, Formal Methods, Proofs Of Programs, Safety, Semantics, Static Analysis

Creation of the Project-Team: 2008 January 01, end of the Project-Team: 2013 December 31.

1. Members

Research Scientists

Xavier Rival [Team leader, Inria, Researcher, HdR]
Patrick Cousot [ENS Paris, Professor on leave of absence in 2013]
Radhia Cousot [CNRS, Senior Researcher (Emeritus)]
Jérôme Feret [Inria, Researcher]
Antoine Miné [CNRS, Researcher, HdR]

PhD Students

Ferdinanda Camporesi [ENS Paris & Bologna University]
Arlen Cox [Inria, granted by European Research Council, from Jul 2013]
Huisong Li [Inria, granted by European Research Council, from Oct 2013]
Mehdi Bouaziz [ENS Paris]
Tie Cheng [Inria, granted by European Research Council]
Antoine Toubhans [ENS Paris]
Caterina Urban [ENS Paris]

Post-Doctoral Fellows

Wassim Abou-Jaoudé [ENS Paris from December 2013]
Norman Ferns [ENS Paris, granted by ANR ABSTRACTCELL project]
Arnaud Spiwack [Inria, granted by European Research Council]

Visiting Scientist

Bor-Yuh Evan Chang [Invited Researcher, granted by European Research Council, from Jul 2013 until Aug 2013]

Administrative Assistant

Marine Meyer [Inria]

Others

Théo Zimmermann [Inria, Undergraduate Intern, granted by European Research Council, from Mar 2013 until May 2013]
Abdellatif Atki [École Polytechnique, M1 Intern, from April 2013 until July 2013]
Matthias Bry [École Polytechnique, M1 Intern, from April 2013 until July 2013]

2. Overall Objectives

2.1. Overall Objectives

Software has known a spectacular development this last decade both in its scope of applicability and its size. Nevertheless, software design, development and engineering methods remain mostly manual, hence error-prone. It follows that complex software-based systems are unsafe and insecure, which is not acceptable in safety-critical or mission-critical applications. Intellectual and computer-based tools must therefore be developed to cope with the safety and security problems.

The notions of *abstraction* and *approximation*, as formalized by the *abstract interpretation theory*, are fundamental to design, model, develop, analyze, and verify highly complex systems, from computer-based to biological ones. They also underlie the design of safety and security verification *tools*.

2.2. Highlights of the Year

Patrick and Radhia Cousot have received in 2013 the SIGPLAN Achievement award, for the invention, development, and application of abstract interpretation http://www.sigplan.org/Awards/Achievement/Main.

3. Research Program

3.1. Abstract Interpretation Theory

The abstract interpretation theory [41], [31], [42], is the main scientific foundation of the work of the ABSTRACTION project-team. Its main current application is on the safety and security of complex hardware and software computer systems either sequential [41], [33], or parallel [35] with shared memory [32], [34], [44] or synchronous message [43] communication.

Abstract interpretation is a theory of sound approximation of mathematical structures, in particular those involved in the behavior of computer systems. It allows the systematic derivation of sound methods and algorithms for approximating undecidable or highly complex problems in various areas of computer science (semantics, verification and proof, model-checking, static analysis, program transformation and optimization, typing, software steganography, etc...) and system biology (pathways analysis).

3.2. Formal Verification by Abstract Interpretation

The *formal verification* of a program (and more generally a computer system) consists in proving that its *semantics* (describing "what the program executions actually do") satisfies its *specification* (describing "what the program executions are supposed to do").

Abstract interpretation formalizes the idea that this formal proof can be done at some level of abstraction where irrelevant details about the semantics and the specification are ignored. This amounts to proving that an *abstract semantics* satisfies an *abstract specification*. An example of abstract semantics is Hoare logic while examples of abstract specifications are invariance, partial, or total correctness. These examples abstract away from concrete properties such as execution times.

Abstractions should preferably be *sound* (no conclusion derived from the abstract semantics is wrong with respect to the program concrete semantics and specification). Otherwise stated, a proof that the abstract semantics satisfies the abstract specification should imply that the concrete semantics also satisfies the concrete specification. Hoare logic is a sound verification method, debugging is not (since some executions are left out), bounded model checking is not either (since parts of some executions are left out). Unsound abstractions lead to *false negatives* (the program may be claimed to be correct/non erroneous with respect to the specification whereas it is in fact incorrect). Abstract interpretation can be used to design sound semantics and formal verification methods (thus eliminating all false negatives).

Abstractions should also preferably be *complete* (no aspect of the semantics relevant to the specification is left out). So if the concrete semantics satisfies the concrete specification this should be provable in the abstract. However program proofs (for non-trivial program properties such as safety, liveness, or security) are undecidable. Nevertheless, we can design tools that address undecidable problems by allowing the tool not to terminate, to be driven by human intervention, to be unsound (e.g. debugging tools omit possible executions), or to be incomplete (e.g. static analysis tools may produce false alarms). Incomplete abstractions lead to *false positives* or *false alarms* (the specification is claimed to be potentially violated by some program executions while it is not). Semantics and formal verification methods designed by abstract interpretation may be complete (e.g. [38], [39], [47]) or incomplete (e.g. [2]).

Sound, automatic, terminating and precise tools are difficult to design. Complete automatic tools to solve non-trivial verification problems cannot exist, by undecidability. However static analysis tools producing very few or no false alarms have been designed and used in industrial contexts for specific families of properties and programs [45]. In all cases, abstract interpretation provides a systematic construction method based on the effective approximation of the concrete semantics, which can be (partly) automated and/or formally verified.

Abstract interpretation aims at:

- providing a basic coherent and conceptual theory for understanding in a unified framework the multiplicity of ideas, concepts, reasonings, methods, and tools on formal program analysis and verification [41], [42];
- guiding the correct formal design of *abstract semantics* [39], [47] and automatic tools for *program analysis* (computing an abstract semantics) and *program verification* (proving that an abstract semantics satisfies an abstract specification) [36].

Abstract interpretation theory studies semantics (formal models of computer systems), abstractions, their soundness, and completeness.

In practice, abstract interpretation is used to design analysis, compilation, optimization, and verification tools which must automatically and statically determine properties about the runtime behavior of programs. For example the ASTRÉE static analyzer (Section 5.2), which was developed by the team over the last decade, aims at proving the absence of runtime errors in programs written in the C programming language. It was originally used in the aerospace industry to verify very large, synchronous, time-triggered, real-time, safety-critical, embedded software and its scope of application was later broadly widened. ASTRÉE is now industrialized by AbsInt Angewandte Informatik GmbH and is commercially available.

3.3. Advanced Introductions to Abstract Interpretation

A short, informal, and intuitive introduction to the theory of abstract interpretation can be found in [36], see also "Abstract Interpretation in a Nutshell" [1] on the web. A more comprehensive introduction is available online [2]. The paper entitled "Basic concepts of abstract interpretation" [37] and an elementary "course on abstract interpretation" [3] can also be found on the web.

4. Application Domains

4.1. Certification of Safety Critical Software

Keywords: Absence of runtime error, Abstract interpretation, Certified compilation, Static analysis, Translation validation, Verifier.

Safety critical software may incur great damage in case of failure, such as human casualties or huge financial losses. These include many kinds of embedded software, such as fly-by-wire programs in aircrafts and other avionic applications, control systems for nuclear power plants, or navigation systems of satellite launchers. For instance, the failure of the first launch of Ariane 5 (flight Ariane 501) was due to overflows in arithmetic computations. This failure caused the loss of several satellites, worth up to $ 500 millions.

This development of safe and secure critical software requires formal methods so as to ensure that they do not go wrong, and will behave as specified. In particular, testing, bug finding methods, checking of models but not programs do not provide any guarantee that no failure will occur, even of a given type such as runtime errors; therefore, their scope is limited for certification purposes. For instance, testing can usually not be performed for *all* possible inputs due to feasibility and cost reasons, so that it does not prove anything about a large number of possible executions.

[1] www.di.ens.fr/~cousot/AI/IntroAbsInt.html
[2] www.di.ens.fr/~cousot/AI/
[3] web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/

By contrast, program analysis methods such as abstract-interpretation-based static analysis are not subject to unsoundness, since they can *formally prove* the absence of bugs directly on the program, not on a model that might be erroneous. Yet, these techniques are generally incomplete since the absence of runtime errors is undecidable. Therefore, in practice, they are prone to false alarms (*i.e.*, they may fail to prove the absence of runtime errors for a program which is safe). The objective of certification is to ultimately eliminate all false alarms.

It should be noted that, due to the size of the critical codes (typically from 100 to 1000 kLOCs), only scalable methods can succeed (in particular, software model checking techniques are subject to state explosion issues). As a consequence, this domain requires efficient static analyses, where costly abstractions should be used only parsimoniously.

Furthermore, many families of critical software have similar features, such as the reliance on floating-point intensive computations for the implementation of control laws, including linear and non-linear control with feedback, interpolations, and other DSP algorithms. Since we stated that a proof of absence of runtime errors is required, very precise analyses are required, which should be able to yield no false alarm on wide families of critical applications. To achieve that goal, significant advantages can be found in the design of domain specific analyzers, such as ASTRÉE [30], [46], which has been initially designed specifically for synchronous embedded software.

Last, some specific critical software qualification procedures may require additional properties being proved. As an example, the DO-178 regulations (which apply to avionics software) require a tight, documented, and certified relation to be established between each development stage. In particular, compilation of high level programs into executable binaries should also be certified correct.

The ABSTRACTION project-team has been working on both proof of absence of runtime errors and certified compilation over the decade, using abstract interpretation techniques. Successful results have been achieved on industrial applications using the ASTRÉE analyzer. Following this success, ASTRÉE has been licensed to AbsInt Angewandte Informatik GmbH to be industrialized, and the ABSTRACTION project-team has strong plans to continue research on this topic.

4.2. Abstraction of Biological Cell Signaling Networks

Keywords: Biology, Health, Static analysis.

Protein-protein interactions consist in complexations and post translational modifications such as phosphorilation. These interactions enable biological organisms to receive, propagate, and integrate signals that are expressed as proteins concentrations in order to make decisions (on the choice between cell division and cell death for instance). Models of such interaction networks suffer from a combinatorial blow up in the number of species (number of non-isomorphic ways in which some proteins can be connected to each others). This large number of species makes the design and the analysis of these models a highly difficult task. Moreover the properties of interest are usually quantitative observations on stochastic or differential trajectories, which are difficult to compute or abstract.

Contextual graph-rewriting systems allow a concise description of these networks, which leads to a scalable method for modeling them. Then abstract interpretation allows the abstraction of these systems properties. First qualitative abstractions (such as over approximation of complexes that can be built) provide both debugging information in the design phases (of models) and static information that are necessary in order to make other computations (such as stochastic simulations) scale up. Then qualitative invariants also drive efficient quantitative abstractions (such as the reduction of ordinary differential semantics).

The work of the ABSTRACTION project-team on biological cell signaling networks ranges from qualitative abstractions to quantitative abstractions.

5. Software and Platforms

5.1. The Apron Numerical Abstract Domain Library

Participants: Antoine Miné [correspondent], Bertrand Jeannet [team PopArt, Inria-RA].

Keywords: Convex polyhedra, Intervals, Linear equalities, Numerical abstract domain, Octagons.

The APRON library is dedicated to the static analysis of the numerical variables of a program by abstract interpretation. Its goal is threefold: provide ready-to-use numerical abstractions under a common API for analysis implementers, encourage the research in numerical abstract domains by providing a platform for integration and comparison of domains, and provide a teaching and demonstration tool to disseminate knowledge on abstract interpretation.

The APRON library is not tied to a particular numerical abstraction but instead provides several domains with various precision versus cost trade-offs (including intervals, octagons, linear equalities and polyhedra). A specific C API was designed for domain developers to minimize the effort when incorporating a new abstract domain: only few domain-specific functions need to be implemented while the library provides various generic services and fallback methods (such as scalar and interval operations for most numerical data-types, parametric reduced products, and generic transfer functions for non-linear expressions). For the analysis designer, the APRON library exposes a higher-level API with C, C++, OCaml, and Java bindings. This API is domain-neutral and supports a rich set of semantic operations, including parallel assignments (useful to analyze automata), substitutions (useful for backward analysis), non-linear numerical expressions, and IEEE floating-point arithmetic.

The APRON library is freely available on the web at http://apron.cri.ensmp.fr/library; it is distributed under the LGPL license and is hosted at InriaGForge. Packages exist for the Debian and Fedora Linux distributions. In order to help disseminate the knowledge on abstract interpretation, a simple inter-procedural static analyzer for a toy language is included. An on-line version is deployed at http://pop-art.inrialpes.fr/interproc/interprocweb.cgi.

The APRON library is developed since 2006 and currently consists of 130 000 lines of C, C++, OCaml, and Java.

Current and past external library users include the Constraint team (LINA, Nantes, France), the Proval/Démon team (LRI Orsay, France), the Analysis of Computer Systems Group (New-York University, USA), the Sierum software analysis platform (Kansas State University, USA), NEC Labs (Princeton, USA), EADS CCR (Paris, France), IRIT (Toulouse, France), ONERA (Toulouse, France), CEA LIST (Saclay, France), VERIMAG (Grenoble, France), ENSMP CRI (Fontainebleau, France), the IBM T.J. Watson Research Center (USA), the University of Edinburgh (UK).

Additionally, APRON is used internally by the team to assist the research on numeric domains and static analyses by enabling the development of fast prototypes. In 2013, APRON has been used to design a sufficient-condition generator prototype (6.3.2), the FUNCTION prototype analyzer for termination (5.6,6.12), a constraint solver based on numeric abstract domains (6.5), a prototype implementation and extension of the Two Variables Per Inequality abstract domain [27].

5.2. The Astrée Static Analyzer of Synchronous Software

Participants: Patrick Cousot [project scientific leader, correspondent], Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival.

Keywords: Absence of runtime error, Abstract interpretation, Static analysis, Verifier.

ASTRÉE is a static analyzer for sequential programs based on abstract interpretation [41], [31], [42], [33].

The ASTRÉE static analyzer [30], [46][1] www.astree.ens.fr aims at proving the absence of runtime errors in programs written in the C programming language.

ASTRÉE analyzes structured C programs, with complex memory usages, but without dynamic memory allocation nor recursion. This encompasses many embedded programs as found in earth transportation, nuclear energy, medical instrumentation, and aerospace applications, in particular synchronous control/command. The whole analysis process is entirely automatic.

ASTRÉE discovers all runtime errors including:

- undefined behaviors in the terms of the ANSI C99 norm of the C language (such as division by 0 or out of bounds array indexing);
- any violation of the implementation-specific behavior as defined in the relevant Application Binary Interface (such as the size of integers and arithmetic overflows);
- any potentially harmful or incorrect use of C violating optional user-defined programming guidelines (such as no modular arithmetic for integers, even though this might be the hardware choice);
- failure of user-defined assertions.

The analyzer performs an abstract interpretation of the programs being analyzed, using a parametric domain (ASTRÉE is able to choose the right instantiation of the domain for wide families of software). This analysis produces abstract invariants, which over-approximate the reachable states of the program, so that it is possible to derive an *over*-approximation of the dangerous states (defined as states where any runtime error mentioned above may occur) that the program may reach, and produces alarms for each such possible runtime error. Thus the analysis is sound (it correctly discovers *all* runtime errors), yet incomplete, that is it may report false alarms (*i.e.*, alarms that correspond to no real program execution). However, the design of the analyzer ensures a high level of precision on domain-specific families of software, which means that the analyzer produces few or no false alarms on such programs.

In order to achieve this high level of precision, ASTRÉE uses a large number of expressive abstract domains, which allow expressing and inferring complex properties about the programs being analyzed, such as numerical properties (digital filters, floating-point computations), Boolean control properties, and properties based on the history of program executions.

ASTRÉE has achieved the following two unprecedented results:

- **A340–300.** In Nov. 2003, ASTRÉE was able to prove completely automatically the absence of any RTE in the primary flight control software of the Airbus A340 fly-by-wire system, a program of 132,000 lines of C analyzed in 1h20 on a 2.8 GHz 32-bit PC using 300 MB of memory (and 50mn on a 64-bit AMD Athlon 64 using 580 MB of memory).
- **A380.** From Jan. 2004 on, ASTRÉE was extended to analyze the electric flight control codes then in development and test for the A380 series. The operational application by Airbus France at the end of 2004 was just in time before the A380 maiden flight on Wednesday, 27 April, 2005.

These research and development successes have led to consider the inclusion of ASTRÉE in the production of the critical software for the A350. ASTRÉE is currently industrialized by AbsInt Angewandte Informatik GmbH and is commercially available.

5.3. The AstréeA Static Analyzer of Asynchronous Software

Participants: Patrick Cousot [project scientific leader, correspondent], Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival.

Keywords: Absence of runtime error, Abstract interpretation, Data races, Interference, Memory model, Parallel software, Static analysis, Verifier.

ASTRÉEA is a static analyzer prototype for parallel software based on abstract interpretation [43], [44], [35]. It started with support from THÉSÉE ANR project (2006–2010) and is continuing within the ASTRÉEA project (2012–2015).

The ASTRÉEA prototype www.astreea.ens.fr is a fork of the ASTRÉE static analyzer (see 5.2) that adds support for analyzing parallel embedded C software.

ASTRÉEA analyzes C programs composed of a fixed set of threads that communicate through a shared memory and synchronization primitives (mutexes, FIFOs, blackboards, etc.), but without recursion nor dynamic creation of memory, threads nor synchronization objects. ASTRÉEA assumes a real-time scheduler, where thread scheduling strictly obeys the fixed priority of threads. Our model follows the ARINC 653 OS specification used in embedded industrial aeronautic software. Additionally, ASTRÉEA employs a weakly-consistent memory semantics to model memory accesses not protected by a mutex, in order to take into account soundly hardware and compiler-level program transformations (such as optimizations). ASTRÉEA checks for the same run-time errors as ASTRÉE, with the addition of data-races.

Compared to ASTRÉE, ASTRÉEA features: a new iterator to compute thread interactions, a refined memory abstraction that takes into account the effect of interfering threads, and a new scheduler partitioning domain. This last domain allows discovering and exploiting mutual exclusion properties (enforced either explicitly through synchronization primitives, or implicitly by thread priorities) to achieve a precise analysis.

ASTRÉEA is currently being applied to analyze a large industrial avionic software: 1.6 MLines of C and 15 threads, completed with a 2,500-line model of the ARINC 653 OS developed for the analysis. The analysis currently takes a few tens of hours on a 2.9 GHz 64-bit intel server using one core and generates around 1,050 alarms. The low computation time (only a few times larger than the analysis time by ASTRÉE of synchronous programs of a similar size and structure) shows the scalability of the approach (in particular, we avoid the usual combinatorial explosion associated to thread interleavings). Precision-wise, the result, while not as impressive as that of ASTRÉE, is quite encouraging. The development of AstréeA continues within the scope of the ASTRÉEA ANR project (8.1.1.2).

5.4. The MemCAD static analyzer

Participants: Xavier Rival [correspondent], Antoine Toubhans.

Keywords: Shape analysis. MemCAD is a static analyzer that focuses on memory abstraction. It takes as input C programs, and computes invariants on the data structures manipulated by the programs. It can also verify memory safety. It comprises several memory abstract domains (flat representation, graph abstraction with summaries based on inductive definitions of data-structures, such as lists) and combination operators for memory abstract domains (hierarchical abstraction, reduced product). The current implementation comes with over 200 small size test cases that are used as regression tests.

5.5. A New Tactic Engine for Coq

Participant: Arnaud Spiwack [Correspondent].

Keywords: Coq, Proof assistant, Dependent types, Tactics, Proof search.

Coq is a proof assistant based on dependent type theory developed chiefly at Inria. This project addresses longstanding usability issues when developing proofs interactively: proofs, in Coq, are typically sequences of instructions – called tactics – which transform the proof into further proof obligations. The expressiveness of tactic affects the kind of proofs which can be written realistically in Coq. Two issues have been addressed. First, providing more backtracking primitives: in a typical automated procedure – which a Coq user could write to discharge proof obligations without human effort – there is some amount of non-determinism. It is hence important to be able to devise strategies, and Coq suffered from limited options on that front. The new tactics support further primitives loosely inspired by Prolog, in particular a backtracking choice (like disjunction in Prolog), and a primitive “once” which is akind to Prolog’s soft-cut control primitive.

The second issue is more fundamental: since Coq is based on dependent types, a proof can appear in the statement of a proof obligation. As a result, tactics should be able to handle so-called dependent subgoals (where several proof obligations are left to be discharged, and the proof of one of them is mentioned into the statement of another). This was not historically the case in Coq, which had a direct influence on some users.

Both of the backtracking primitive and the dependent subgoals are part of the development version of Coq and will be part of the next release.

5.6. FuncTion: An Abstract Domain Functor for Termination

Participant: Caterina Urban.

Keywords: Conditional termination, Ranking functions, Static analysis.

FUNCTION is a research prototype static analyzer to analyze the termination of programs written in a small non-deterministic imperative language: it can infer sufficient conditions so that all executions terminate (conditional definitive termination). Following the general framework to analyze termination by abstract interpretation proposed in [40], FUNCTION infers ranking functions using piecewise-defined abstract domains. A first version of FUNCTION implemented a domain of linear ranking functions partitioned by variable bounds [24], [23]. It has been generalized in [22] and [29] to support ordinal-valued ranking functions of the form $\sum_{i=1}^{n} \omega^i f_i$ where n is a constant and each f_i is a natural-valued linear function of the variables.

The analyzer is written in OCaml and implemented on top of the APRON library (5.1). It can be used on-line through a web interface: http://www.di.ens.fr/~urban/FuncTion.html.

FUNCTION entered the 3rd Competition on Software Verification (SV-COMP 2014) in the termination category (demonstration section, no ranking).

5.7. The OpenKappa Modeling Plateform

Participants: Monte Brown [Harvard Medical School], Vincent Danos [University of Edinburgh], Jérôme Feret [Correspondent], Luca Grieco, Walter Fontana [Harvard Medical School], Russ Harmer [ENS Lyon], Jean Krivine [Paris VII].

Keywords: Causal traces, Model reduction, Rule-based modeling, Simulation, Static analysis.

OPENKAPPA is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language [53], a static analyzer [52] (for debugging models), a simulator [51], a compression tool for causal traces [50], [48], and a model reduction tool [4], [49], [58].

OPENKAPPA is developed since 2007 and, the OCaml version currently consists of 46 000 lines of OCaml. Software are available in OCaml and in Java. Moreover, an Eclipse pluggin is available.A compiler from CellDesigner into Kappa has been released in 2013.

OPENKAPPA is freely available on the web at http://kappalanguage.org under the LGPL license. Discussion groups are also available on line.

Current external users include the ETH Zürich, the UNAM-Genomics Mexico team. It is used as pedagocical material in graduate lessons at Harvard Medical School, and at the Interdisciplinary Approaches to Life science (AIV) Master Program (Université de Médecine Paris-Descartes).

5.8. Translation Validation

Participant: Xavier Rival [correspondent].

Keywords: Abstract interpretation, Certified compilation, Static analysis, Translation validation, Verifier.

The main goal of this software project is to make it possible to certify automatically the compilation of large safety critical software, by proving that the compiled code is correct with respect to the source code: When the proof succeeds, this guarantees that no compiler bug did cause incorrect code to be generated. Furthermore, this approach should allow to meet some domain specific software qualification criteria (such as those in DO-178 regulations for avionics software), since it allows proving that successive development levels are correct with respect to each other *i.e.*, that they implement the same specification. Last, this technique also justifies the use of source level static analyses, even when an assembly level certification would be required, since it establishes separately that the source and the compiled code are equivalent.

The compilation certification process is performed automatically, thanks to a prover designed specifically. The automatic proof is done at a level of abstraction which has been defined so that the result of the proof of equivalence is strong enough for the goals mentioned above and so that the proof obligations can be solved by efficient algorithms.

The current software features both a C to Power-PC compilation certifier and an interface for an alternate source language frontend, which can be provided by an end-user.

5.9. Zarith

Participants: Antoine Miné [Correspondent], Xavier Leroy [Inria Paris-Rocquencourt], Pascal Cuoq [CEA LIST].

Keywords: Arbitrary precision integers, Arithmetic, OCaml.

ZARITH is a small (10K lines) OCaml library that implements arithmetic and logical operations over arbitrary-precision integers. It is based on the GNU MP library to efficiently implement arithmetic over big integers. Special care has been taken to ensure the efficiency of the library also for small integers: small integers are represented as Caml unboxed integers and use a specific C code path. Moreover, optimized assembly versions of small integer operations are provided for a few common architectures.

ZARITH is an open-source project hosted at OCamlForge (http://forge.ocamlcore.org/projects/zarith) and distributed under a modified LGPL license.

ZARITH is currently used in the ASTRÉE analyzer to enable the sound analysis of programs featuring 64-bit (or larger) integers. It is also used in the Frama-C analyzer platform developed at CEA LIST and Inria Saclay.

6. New Results

6.1. Analysis of Biological Pathways

We have improved our framework to design and analyze biological networks in KAPPA. This framework focuses on protein-protein interaction networks described as graph rewriting systems. Such networks can be used to model some signaling pathways that control the cell cycle. The task is made difficult due to the combinatorial blow up in the number of reachable species (*i.e.*, non-isomorphic connected components of proteins).

6.1.1. Semantics

Participants: Jonathan Hayman, Tobias Heindel [CEA-List].

Keywords: Graph rewriting, Single Push-Out semantics.

Domain-specific rule-based languages can be understood intuitively as transforming graph-like structures, but due to their expressivity these are difficult to model in ‘traditional’ graph rewriting frameworks.

In [16], we introduce pattern graphs and closed morphisms as a more abstract graph-like model and show how Kappa can be encoded in them by connecting its single-pushout semantics to that for Kappa. This level of abstraction elucidates the earlier single-pushout result for Kappa, teasing apart the proof and guiding the way to richer languages, for example the introduction of compartments within cells.

6.1.2. Causality Analysis

We use causal analysis so as to extract minimal concurrent scenarios that lead to the activation of given events.

6.1.2.1. Implementation

Participant: Jérôme Feret.

Keywords: Causality, Counter-examples, Compression.

This year, we have re-implemented in OPENKAPPA the strong compression method that is described in [48]. The new implementation is very efficient, it has been used to extract minimal scenarios from traces of several hundred of thousands causally related events, that were generated during the simulation of a model of the WnT signaling pathway.

6.1.2.2. Framework

Participant: Jonathan Hayman.

Keywords: Abstraction, Causality, Compression.

Standard notions of independence of rule applications fail to provide adequately concise causal histories, leading to the earlier formulation of strong and weak forms of trajectory compression for Kappa. In [15], we give a simple categorical account of how forms of compression can be uniformly obtained. This generalisation also describes a way for the user to specify their own levels of compression between weak and strong, which we call filtered compression. This is based on the idea of the user specifying the part of the type graph that represents the the structure which the compression technique should track through the trace.

6.1.3. Model Reduction

Participants: Ferdinanda Camporesi, Jérôme Feret, Jonathan Hayman.

Keywords: Context-sensitivity, Differential semantics, Model reduction.

Rule-based modeling allows very compact descriptions of protein-protein interaction networks. However, combinatorial complexity increases again when one attempts to describe formally the behaviour of the networks, which motivates the use of abstractions to make these models more coarse-grained. Context-insensitive abstractions of the intrinsic flow of information among the sites of chemical complexes through the rules have been proposed to infer sound coarse-graining, providing an efficient way to find macro-variables and the corresponding reduced models.

In [12], we propose a framework to allow the tuning of the context-sensitivity of the information flow analyses and show how these finer analyses can be used to find fewer macro-variables and smaller reduced differential models.

6.2. Andromeda: Accurate and Scalable Security Analysis of Web Applications

Participants: Omer Tripp [Tel Aviv University, Israël], Marco Pistola [University of Washington, Seattle, USA], Patrick Cousot, Radhia Cousot, Salvatore Guarnieri.

Keywords: Abstract interpretation, Security, Web.

Security auditing of industry-scale software systems mandates automation. Static taint analysis enables deep and exhaustive tracking of suspicious data flows for detection of potential leakage and integrity violations, such as cross-site scripting (XSS), SQL injection (SQLi) and log forging. Research in this area has taken two directions: program slicing and type systems. Both of these approaches suffer from a high rate of false findings, which limits the usability of analysis tools based on these techniques. Attempts to reduce the number of false findings have resulted in analyses that are either (i) unsound, suffering from the dual problem of false negatives, or (ii) too expensive due to their high precision, thereby failing to scale to real-world applications.

In [21], we investigate a novel approach for enabling precise yet scalable static taint analysis. The key observation informing our approach is that taint analysis is a demand-driven problem, which enables lazy computation of vulnerable information flows, instead of eagerly computing a complete data-flow solution, which is the reason for the traditional dichotomy between scalability and precision. We have implemented our approach in Andromeda, an analysis tool that computes data-flow propagations on demand, in an efficient and accurate manner, and additionally features incremental analysis capabilities. Andromeda is currently in use in a commercial product. It supports applications written in Java, .NET and JavaScript. Our extensive evaluation of Andromeda on a suite of 16 production-level benchmarks shows Andromeda to achieve high accuracy and compare favorably to a state-of-the-art tool that trades soundness for precision.

6.3. Backward analysis

6.3.1. Automatic Inference of Necessary Preconditions

Participants: Patrick Cousot, Radhia Cousot, Manuel Fähndrich [Microsoft Research, Redmond, USA], Francesco Logozzo [Microsoft Research, Redmond, USA].

Keywords: Abstract interpretation, Backward analysis, Static analysis, Necessary condition inference.

In [14], we consider the problem of automatic precondition inference for: (i) program verification; (ii) helping the annotation process of legacy code; and (iii) helping generating code contracts during code refactoring. We argue that the common notion of sufficient precondition inference (i.e., under which precondition is the program correct?) imposes too large a burden on call-sites, and hence is unfit for automatic program analysis. Therefore, we define the problem of necessary precondition inference (i.e., under which precondition, if violated, will the program always be incorrect?). We designed and implemented several new abstract interpretation-based analyses to infer necessary preconditions. The analyses infer atomic preconditions (including disjunctions), as well as universally and existentially quantified preconditions.

We experimentally validated the analyses on large scale industrial code.

For unannotated code, the inference algorithms find necessary preconditions for almost 64% of methods which contained warnings. In 27% of these cases the inferred preconditions were also sufficient, meaning all warnings within the method body disappeared. For annotated code, the inference algorithms find necessary preconditions for over 68% of methods with warnings. In almost 50% of these cases the preconditions were also sufficient. Overall, the precision improvement obtained by precondition inference (counted as the additional number of methods with no warnings) ranged between 9% and 21%.

6.3.2. Under-approximations to infer sufficient program conditions

Participant: Antoine Miné.

Keywords: Abstract interpretation, Backward analysis, Numerical abstract domains, Static analysis, Sufficient condition inference, Under-approximations.

In [9] we discuss the automatic inference of sufficient preconditions by abstract interpretation and sketch the construction of an under-approximating backward analysis. We focus on numeric properties of variables and revisit three classic numeric abstract domains: intervals, octagons, and polyhedra, with new under-approximating backward transfer functions, including the support for non-deterministic expressions, as well as lower widenings to handle loops. We show that effective under-approximation is possible natively in these domains without necessarily resorting to disjunctive completion nor domain complementation. Applications include the derivation of sufficient conditions for a program to never step outside an envelope of safe states, or dually to force it to eventually fail. We built a proof-of-concept prototype implementation based on the APRON numeric domain library and experimented it on simple examples (the prototype is available for download and usable on-line at http://www.di.ens.fr/~mine/banal).

6.4. Bisimulation metrics

6.4.1. Bisimulation for MDP through Families of Functional Expressions

Participants: Norman Ferns, Sophia Knight [LIX], Doina Precup [McGill University].

Keywords: Markov decision processes, Bisimulation, Metrics.

We have transfered a notion of quantitative bisimilarity for labelled Markov processes [54] to Markov decision processes with continuous state spaces. This notion takes the form of a pseudometric on the system states, cast in terms of the equivalence of a family of functional expressions evaluated on those states and interpreted as a real-valued modal logic. Our proof amounts to a slight modification of previous techniques [61], [60] used to prove equivalence with a fixed-point pseudometric on the state-space of a labelled Markov process and making heavy use of the Kantorovich probability metric. Indeed, we again demonstrate equivalence with a fixed-point pseudometric defined on Markov decision processes [57]; what is novel is that we recast this proof in terms of integral probability metrics [59] defined through the family of functional expressions, shifting emphasis back to properties of such families. The hope is that a judicious choice of family might lead to something more computationally tractable than bisimilarity whilst maintaining its pleasing theoretical guarantees. Moreover, we use a trick from descriptive set theory to extend our results to MDPs with bounded measurable reward functions, dropping a previous continuity constraint on rewards and Markov kernels.

This work is under submission.

6.4.2. Bisimulation Metrics are Optimal Value Functions

Participants: Norman Ferns, Doina Precup [McGill University].

Keywords: Markov decision processes, Bisimulation, Metrics.

We have proved that a behavioural pseudometric defined on the state space of a given Markov decision process and whose kernel is stochastic bisimilarity [57] can be expressed as the optimal value function of another Markov decision process. Furthermore, this latter process can be interpreted as an optimal coupling of two copies of the original model.

This work is under submission.

6.5. A Constraint Solver Based on Abstract Domains

Participants: Marie Pelleau [University of Nantes, LINA], Antoine Miné, Charlotte Truchet [University of Nantes, LINA], Frédéric Benhamou [University of Nantes, LINA].

Keywords: Abstract interpretation, Backward analysis, Numerical abstract domains, Static analysis, Sufficient condition inference, Under-approximations.

In [18] and [19] we apply techniques from abstract interpretation to constraint programming (which aims at solving hard combinatorial problems with a generic framework based on first-order logics). We highlight some links and differences between these fields: both compute fixpoints by iterations but employ different extrapolation and refinement strategies; moreover, consistencies in constraint programming can be mapped to non-relational abstract domains. We then use these correspondences to build an abstract constraint solver that leverages abstract interpretation techniques (such as relational domains) to go beyond classic solvers. We present encouraging experimental results obtained with our prototype implementation.

6.6. A Galois Connection Calculus for Abstract Interpretation

Participants: Patrick Cousot, Radhia Cousot.

Keywords: Abstract interpretation, Galois connection.

In [10], we introduce a Galois connection calculus for language independent specification of abstract interpretations used in programming language semantics, formal verification, and static analysis. This Galois connection calculus and its type system are typed by abstract interpretation.

6.7. Mechanically Verifying a Shape Analysis

Participant: Arnaud Spiwack.

Keywords: Program verification, Abstract interpretation, Static analysis, Shape analysis, Coq.

The result of a static analysis is only as good as the trust put into its correctness. For critical software, the standards are very high, and trusting a complex tool requires costly inspection of its implementation. Mechanically proving the correctness of static analysers is a way to lower these costs: the exigence of trust is moved from various complex dedicated tools to a single simpler general purpose one.

In this context, Arnaud Spiwack worked on an ongoing Coq implementation and certification of a shape abstract domain. The implementation, named Cosa, is based on Evan Chang and Xavier Rival's Xisa. It targets an intermediary language of Xavier Leroy's Compcert C, and interfaces with the domains of the Verasco project.

The development of Cosa lead Arnaud Spiwack to express the abstract interpretation correctness property in term of refinement calculus, which allowed to use interaction structures (a type theoretic variant of the refinement calculus) as a central structuring element of Cosa. Arnaud Spiwack started investigating how the technology of nominal sets could be leveraged to prove the correctness of unfolding (which involves choosing new names) in Cosa.

6.8. Modular Construction of Shape-Numeric Analyzers

Participants: Bor-Yuh Evan Chang [University of Colorado, Boulder, USA], Xavier Rival.

Keywords: Abstract interpretation, Memory abstraction, Shape abstract domains.

In [13], we discuss the modular construction of memory abstract domains.

The aim of static analysis is to infer invariants about programs that are precise enough to establish semantic properties, such as the absence of run-time errors. Broadly speaking, there are two major branches of static analysis for imperative programs. Pointer and *shape* analyses focus on inferring properties of pointers, dynamically-allocated memory, and recursive data structures, while *numeric* analyses seek to derive invariants on numeric values. Although simultaneous inference of shape-numeric invariants is often needed, this case is especially challenging and is not particularly well explored. Notably, simultaneous shape-numeric inference raises complex issues in the design of the static analyzer itself.

In this paper, we study the construction of such shape-numeric, static analyzers. We set up an abstract interpretation framework that allows us to reason about simultaneous shape-numeric properties by combining shape and numeric abstractions into a modular, expressive abstract domain. Such a modular structure is highly desirable to make its formalization and implementation easier to do and get correct. To achieve this, we choose a concrete semantics that can be abstracted step-by-step, while preserving a high level of expressiveness. The structure of abstract operations (i.e., transfer, join, and comparison) follows the structure of this semantics. The advantage of this construction is to divide the analyzer in modules and functors that implement abstractions of distinct features.

6.9. Reduced Product Combination of Abstract Domains for Shapes

Participants: Bor-Yuh Evan Chang [University of Colorado, Boulder, USA], Xavier Rival, Antoine Toubhans.

Keywords: Abstract interpretation, Memory abstraction, Shape abstract domains.

In [20], we discuss the construction of shape abstract domains by reduced product.

Real-world data structures are often enhanced with additional pointers capturing alternative paths through a basic inductive skeleton (e.g., back pointers, head pointers). From the static analysis point of view, we must obtain several interlocking shape invariants. At the same time, it is well understood in abstract interpretation design that supporting a separation of concerns is critically important to designing powerful static analyses. Such a separation of concerns is often obtained via a reduced product on a case-by-case basis. In this paper, we lift this idea to abstract domains for shape analyses, introducing a domain combination operator for memory abstractions. As an example, we present *simultaneous separating shape graphs*, a product construction that combines instances of separation logic-based shape domains. The key enabler for this construction is a static analysis on inductive data structure definitions to derive relations between the skeleton and the alternative paths. From the engineering standpoint, this construction allows each component to reason independently about different aspects of the data structure invariant and then separately exchange information via a reduction operator. From the usability standpoint, we enable describing a data structure invariant in terms of several inductive definitions that hold simultaneously.

6.10. Relational Thread-Modular Static Value Analysis

Participant: Antoine Miné.

Keywords: Abstract interpretation, Concurrency, Embedded software, Rely-guarantee methods, Run-time errors, Safety.

We study in [17] thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety properties. We then show that previous analyses based on non-relational interferences can be retrieved as coarse computable abstractions of this semantics; additionally, we present novel abstraction examples exploiting our ability to reason more precisely about interferences, including domains to infer relational lock invariants and the monotonicity of counters. Our method and domains have been implemented in the ASTRÉEA static analyzer (5.3) that checks for run-time errors in embedded concurrent C programs, where they enabled a significant reduction of the number of false alarms.

6.11. Static Analyzers on the Cloud

Participants: Michael Barnett [Microsoft Research, Redmond, USA], Mehdi Bouaziz, Francesco Logozzo [Microsoft Research, Redmond, USA], Manuel Fähndrich [Microsoft Research, Redmond, USA].

A cloud-based static analyzer runs as service. Clients issue analysis requests through the local network or over the internet. The analysis takes advantage of the large computation resources offered by the cloud: the underlying infrastructure ensures scaling and unlimited storage. Cloud-based analyzers may relax performance-precision trade-offs usually associated with desktop-based analyzers. More cores enable more precise and responsive analyses. More storage enables perfect caching of the analysis results, shareable among different clients, and queryable off-line. To realize these advantages, cloud-based analyzers need to be architected differently than desktop ones. In [11], we describe our ongoing effort of moving a desktop analyzer, Clousot, into a cloud-based one, Cloudot.

6.12. Termination

We have explored the analysis of program termination and the inference of sufficient conditions to ensure the definite termination of programs using abstract interpretation techniques. Following [40], we employ a backward analysis over an abstract domain of ranking functions.

6.12.1. Abstract Domain of Segmented Ranking Functions

Participant: Caterina Urban.

We present in [24] and [23] a parameterized abstract domain that infers sufficient conditions for program termination by automatically synthesizing piecewise-defined ranking functions over natural numbers. The analysis uses over-approximations but we prove its soundness, meaning that all program executions respecting these sufficient conditions are indeed terminating. The abstract domain is parameterized by a numerical abstract domain for environments and a numerical abstract domain for functions. This parameterization allows to easily tune the trade-off between precision and cost of the analysis. We describe an instantiation of this generic domain with intervals and affine functions. We define all abstract operators, including widening to ensure convergence. To experiment with this domain, we have implemented a research prototype static analyzer FUNCTION (5.6) that yielded interesting preliminary results.

6.12.2. Abstract Domain to Infer Ordinal-Valued Ranking Functions

Participants: Caterina Urban, Antoine Miné.

We observed that, in some important cases (such as programs with unbounded non-determinism), there does not exist any ranking function over natural numbers. In [22] and [29] we propose a new abstract domain to automatically infer ranking functions over ordinals. We extended the domain of piecewise-defined natural-valued ranking functions introduced in the previous section to polynomials in ω, where the polynomial coefficients are natural-valued functions of the program variables. The abstract domain is parametric in the choice of the maximum degree of the polynomial, and the types of functions used as polynomial coefficients. We have enriched the FUNCTION prototype analyzer (5.6) with an instantiation of our domain using affine functions as polynomial coefficients. We successfully analyzed small but intricate examples that are out of the reach of existing methods. To our knowledge this is the first abstract domain able to reason about ordinals. Handling ordinals leads to a powerful approach for proving termination of imperative programs, which in particular subsumes existing techniques based on lexicographic ranking functions.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

7.1.1. License agreement

7.1.1.1. Astrée

In February 2009 was signed an exploitation license agreement between CNRS, École Normale Supérieure, and AbsInt Angewandte Informatik GmbH for the industrialization of the ASTRÉE analyzer. ASTRÉE is commercially available from AbsInt since January 2010. Continuous work goes on to adapt the ASTRÉE static analyzer to industrial needs, in particular for the automotive industry. Radhia Cousot is the scientific contact.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. AbstractCell

Title: Formal abstraction of quantitative semantics for protein-protein interaction cellular network models

Instrument: ANR-Chair of Excellence (Junior, long term)

Duration: December 2009 - December 2013

Coordinator: Inria (France)

Others partners: None

See also: http://www.di.ens.fr/ feret/abstractcell

Abstract: The overall goal of this project is to investigate formal foundations and computational aspects of both the stochastic and differential approximate semantics for rule-based models. We want to relate these semantics formally, then we want to design sound approximations for each of these semantics (by abstract interpretation) and investigate scalable algorithms to compute the properties of both the stochastic and the differential semantics. Jérôme Feret is the principal investigator for this project.

8.1.1.2. AstréeA

Title: Static Analysis of Embedded Asynchronous Real-Time Software

Type: ANR Ingénierie Numérique Sécurité 2011

Instrument: ANR grant

Duration: January 2012 - December 2015

Coordinator: Airbus France (France)

Others partners: École normale supérieure (France)

See also: http://www.astreea.ens.fr

Abstract: The focus of the ASTRÉEA project is on the development of static analysis by abstract interpretation to check the safety of large-scale asynchronous embedded software. During the THÉSÉE ANR project (2006–2010), we developed a concrete and abstract models of the ARINC 653 operating system and its scheduler, and a first analyzer prototype. The gist of the ASTRÉEA project is the continuation of this effort, following the recipe that made the success of ASTRÉE: an incremental refinement of the analyzer until reaching the zero false alarm goal. The refinement concerns: the abstraction of process interactions (relational and history-sensitive abstractions), the scheduler model (supporting more synchronisation primitives and taking priorities into account), the memory model (supporting volatile variables), and the abstraction of dynamical data-structures (linked lists). Patrick Cousot is the principal investigator for this project.

8.1.1.3. Verasco

Title: Formally-verified static analyzers and compilers

Type: ANR Ingénierie Numérique Sécurité 2011

Instrument: ANR grant

Duration: Septembre 2011 - September 2015

Coordinator: Inria (France)

Others partners: Airbus France (France), IRISA (France), Inria Saclay (France)

See also: http://www.systematic-paris-region.org/fr/projets/verasco

Abstract: The usefulness of verification tools in the development and certification of critical software is limited by the amount of trust one can have in their results. A first potential issue is *unsoundness* of a verification tool: if a verification tool fails (by mistake or by design) to account for all possible executions of the program under verification, it can conclude that the program is correct while it actually misbehaves when executed. A second, more insidious, issue is *miscompilation*: verification tools generally operate at the level of source code or executable model; a bug in the compilers and code generators that produce the executable code that actually runs can lead to a wrong executable being generated from a correct program.

The project VERASCO advocates a mathematically-grounded solution to the issues of formal verifying compilers and verification tools. We set out to develop a generic static analyzer based on abstract interpretation for the C language, along with a number of advanced abstract domains and domain combination operators, and prove the soundness of this analyzer using the Coq proof assistant. Likewise, we will continue our work on the CompCert C formally-verified compiler, the first realistic C compiler that has been mechanically proved to be free of any miscompilation will be continued. Finally, the tool qualification issues that must be addressed before formally-verified tools can be used in the aircraft industry, will be investigated.

8.2. European Initiatives

8.2.1. FP7 Projects

8.2.1.1. MemCad

Type: IDEAS

Defi: Design Composite Memory Abstract Domains

Instrument: ERC Starting Grant

Objectif: Design Composite Memory Abstract Domains

Duration: October 2011 - September 2016

Coordinator: Inria (France)

Partner: None

Inria contact: Xavier Rival

Abstract: The MemCAD project aims at setting up a library of abstract domains in order to express and infer complex memory properties. It is based on the abstract interpretation frameworks, which allows to combine simple abstract domains into complex, composite abstract domains and static analyzers. While other families of abstract domains (such as numeric abstract domains) can be easily combined (making the design of very powerful static analyses for numeric intensive applications possible), current tools for the analysis of programs manipulating complex abstract domains usually rely on a monolithic design, which makes their design harder, and limits their efficiency. The purpose of the MemCAD project is to overcome this limitation.

Our proposal is based on the observation that the complex memory properties that need to be reasoned about should be decomposed in combinations of simpler properties. Therefore, in static analysis, a complex memory abstract domain could be designed by combining many simpler domains, specific to common memory usage patterns. The benefit of this approach is twofold: first it would make it possible to simplify drastically the design of complex abstract domains required to reason about complex softwares, hereby allowing certification of complex memory intensive softwares by automatic static analysis; second, it would enable to split down and better control the cost of the analyses, thus significantly helping scalability. As part of this project, we propose to build a static analysis framework for reasoning about memory properties, and put it to work on important classes of applications, including large softwares.

8.3. International Initiatives

8.3.1. Informal International Partners

Research on Kappa and its applications involves several close international partners:

- Vincent Danos (University of Edinburgh, UK);
- Walter Fontana (Harvard Medical School, US);
- Hein Koeppl and Tatjana Petrov (ETH Zürich, SW);
- Jonathan Hayman and Glynn Winskel (Cambridge, UK).

Research on abstract domains for memory states involves the group of Bor-Yuh Evan Chang (University of Colorado at Boulder, Colorado, USA).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

Bor-Yuh Evan Chang visited the team from June to August 2013, as part of his collaboration with Xavier Rival.

8.4.1.1. Internships

Abdellatif Atki is a student at École Polytechnique (Palaiseau, France). He performed his M1 internship from April 2013 to July 2013 under the supervision of Antoine Miné on the Two variables per inequality abstract domain [27].

Matthias Bry is a student at École Polytechnique (Palaiseau, France). He performed his M1 internship from April 2013 to July 2013 under the supervision of Antoine Miné on analysis of concurrent programs [28].

Huisong Li is a master student at the Institute of Software, at the Chinese Accademy of Sciences (Beijing, China) and is doing a research internship under the supervision of Xavier Rival.

8.4.2. Visits to International Teams

Xavier Rival visited the ROSAEC Team in Seoul National University (team of Professor Kwangkeun Yi).

9. Dissemination

9.1. Scientific Animation

9.1.1. Academy Members, Professional Societies

Patrick Cousot is a member of the Academia Europaea.

Patrick Cousot is member of the IFIP working group WG 2.3 on programming methodology.

Patrick Cousot is a member of the Board of Trustees and of the Scientific Advisory Board of the IMDEA(Instituto madrileño de estudios avanzados—Research Institute in Software Development Technology), Madrid, Spain and of the Asian Association for Foundations of Software (AAFS).

9.1.2. Collective Responsibilities

Jérôme Feret and Xavier Rival are members of the lab council of the Laboratoire d'Informatique de l'École normale supérieure.

Jérôme Feret was a member of the *comité de sélection* (hiring committee) to hire an assistant professor at the Université de Lille 1.

Antoine Miné was a member of the *comité de sélection* (hiring committee) to hire an assistant professor at the École normale supérieure (Paris).

9.1.3. Editorial Boards and Program Committees

— Patrick Cousot is member of the advisory board of the Higher-Order Symbolic Computation journal (HOSC, Springer) and of the Journal of Computing Science and Engineering (JCSE, Kiise).

Patrick Cousot is member of the steering committees of the Static Analysis Symposium (SAS) and the Verification, Model-Checking and Abstract Interpretation (VMCAI) international conference.

— Radhia Cousot is member of the advisory board of the Higher-Order Symbolic Computation journal (HOSC, Springer) and the Central European Journal of Computer Science (CEJCS, Versita & Springer).

Radhia Cousot is member of the steering committees of the Static Analysis Symposium (SAS), the Workshop on Numerical and Symbolic Abstract Domains (NSAD), the Workshop on Static Analysis and Systems Biology (SASB) and the Workshop on Tools for Automatic Program AnalysiS (TAPAS).

Radhia Cousot was the program committee chair of the 40th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 2013), Rome, Italy, January 23-25, 2013 [25].

Radhia Cousot was member of the program committee of the 5th NASA Formal Methods Symposium (NFM 2013), May 13-16, 2013, NASA Ames Research Center, California, USA.

— Jérôme Feret is a member of the editorial board of the Frontiers in Genetics journal and the Open Journal of Modelling and Simulation

Jérôme Feret is a member of the steering committee of the Workshop on Static Analysis and Systems Biology (SASB).

Jérôme Feret was co-program committee chair of the 4th International Workshop on Static Analysis and Systems Biology (SASB 2013), Seattle, USA, June 19, 2013.

Jérôme Feret was member of the program committees of the 40th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2013 ERC), Rome, Italy, January 23-25, 2013; the 5th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2013), Lisbon, Portugal, March 24-29, 2013; the 4th International Workshop on Computational Models for Cell Processes (CompMod 2013), Turku, Finland, June 11, 2013; the 11th Conference on Computational Methods in Systems Biology (CMSB 2013), Klosterneuburg, Austria, September 23-25, 2013; the 2nd International Conference on Biomedical Engineering and Biotechnology (ICBEB 2013), 2013; the 15th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2014), San Diego, USA, January 19-21, 2014. He is member of the program committees of the 6th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2014), Chamonix, France, April 20-24, 2014; the 10th International Workshop on Developments in Computational Models (DCM 2014), Vienna, Austria, July 13, 2014; the International Workshop on Verification of Molecular Devices and Programs (VEMDP 2014), Vienna, Austria, July 17, 2014; the 8th IFIP Theoretical Computer Science Conference (IFIP TCS 2014), Rome, Italy, September 1-3, 2014; the 5th International Workshop on Static Analysis and Systems Biology (SASB 2014), Munich, Germany, September 10, 2014; the 9th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2014), Florence, Italy, September 11-12, 2014; the 3rd International Conference on Biomedical Engineering and Biotechnology (ICBEB 2014), Beijing, China, September 19-21, 2014.

— Jonathan Hayman was a member of the program committees of the 40th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2013 ERC), Rome, Italy, January 23-25, 2013; the 4th International Workshop on Static Analysis and Systems Biology (SASB 2013), Seattle, USA, June 19, 2013.

— Antoine Miné is a member of the editorial boards of The Scientific World Journal in Computer Science and of the journal Conference Papers in Computer Science.

Antoine Miné was a member of the program committee of the 20th International Static Analysis Symposium (SAS 2013), Seattle, WA, USA, June 20–22, 2013, and the 3rd International Workshop on Safety and Security in Cyber-Physical Systems (SSCPS 2013), Washington, D.C, USA., June, 18–20, 2013. He is a member of the program committee of the 11th School on Modelling and Verifying Parallel Processes (MOVEP 2014), Nantes, France, July 7–11, 2014, the 8th International Symposium on Theoretical Aspects of Software Engineering (TASE 2014), Changsha, China, 1–3 September, 2014, the 11th International Conference on Integrated Formal Methods (iFM 2014), Bertinoro, Italy, September 9–11, 2014, and of the external review committee of the 42nd ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2015 ERC), Mumbai, India, January 11–18, 2015.

— Xavier Rival is member of the steering committee of the Workshop on Tools for Automatic Program AnalysiS (TAPAS).

Xavier Rival is Co-Chair of the program committee of the 15th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2014, San Diego, USA, January 19-21, 2014 [26].

Xavier Rival was member of the extended review committee of the 40th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2013 ERC), Rome, Italy, January 23-25, 2013; the program committee of the program committee the European Symposium On Programming (ESOP 2013), Rome, Italy, March 2013; the program committee of the 41st ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2014), San Diego, USA, January 22-24, 2014;

9.1.4. Participation in Conferences

Luminy workshop: Workshop on Modelisation, Optimisation, and Static Analysis (Luminy, France, January 6–10, 2013).
Jérôme Feret attended the workshop and gave a talk on model reduction of Kappa models.

VMCAI: 14th International Conference on Verification, Model Checking, and Abstract Interpretation (Roma, Italy, January 20–22, 2013).
Antoine Miné, Xavier Rival, Antoine Toubhans and Caterina Urban attended the conference. Antoine Toubhans presented an article [20].

POPL: 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (Roma, Italy, January 23–25, 2013).
Xavier Rival and Caterina Urban attended the conference.

Bytecode: Eigth Workshop on Bytecode Semantics, Verification, Analysis, and Transformation (Rome, Italy, March 23, 2013).
Mehdi Bouaziz attended the workshop and gave a talk on [11].

CC: Twenty-second International Conference on Compiler Construction (Rome, Italy, March 21–22, 2013).
Mehdi Bouaziz attended the conference.

ESOP: Twenty-second European Symposium on Programming (Rome, Italy, March 19–22, 2013).
Mehdi Bouaziz attended the conference.

FASE: Sixteenth International Conference on Fundamental Approaches to Software Engineering(Rome, Italy, March 20–22, 2013).
Mehdi Bouaziz attended the conference.

POST: Second Conference on Principles of Security and Trust (Rome, Italy, March 18–19, 2013).
Mehdi Bouaziz attended the conference.

TACAS: Nineteenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Rome, Italy, March 18–21, 2013).
Mehdi Bouaziz attended the conference.

RBMW: Rule-Based Modelling Workshop (Paris, France, April 15–16, 2013).
Wassim Abou-jaoudé, Ferdinanda Camporesi, Jérôme Feret, and Norman Ferns attended the workshop. Jérôme Feret gave a talk on [12]. Norman Ferns gave a talk on backward bisimulation.

In'Tech: In'Tech Seminar on Formal validation of industrial critical systems (Grenoble, France, April 18, 2013).
Jérôme Feret attended the workshop and gave a talk on the ASTRÉE analyzer.

Dagstuhl 13162: Wokshop on Pointer Analysis (Dagstuhl, Germany, April 14–19, 2013).
Xavier Rival attended the workshop and gave a talk on the modular construction of shape analyzers.

SASB: Fourth International Workshop on Static Analysis and Systems Biology (Seattle, USA, June 19, 2013).
Jérôme Feret and Jonathan Hayman attended the workshop. Jérôme Feret chaired the workshop and chaired half of the sessions. Jérôme Feret gave a talk on [12] and Jonathan Hayman gave a talk on [15].

SAS: 21th International Static Analysis Symposium (Seattle, WA, USA, June 20–22, 2013).
Jérôme Feret and Caterina Urban attended the conference, and Caterina Urban presented a paper [24].

ForumMF: Deuxième Forum Méthodes Formelles, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS, Toulouse, France, June 28, 2013).
Mehdi Bouaziz attended the workshop and gave a talk on Specification and verification of programs with CodeContracts.

WST: 13th International Workshop on Termination (Bertinoro, Italy, 29–31 August, 2013).
Caterina Urban attended the workshop and presented an article [23].

DSS2013: Dave Schmidt Symposium (Festschrift for Dave Schmidt, Manhattan, Kansas, September 18-19, 2013).
Xavier Rival attended the symposium and presented an article [13].

In'Tech: In'Tech Seminar on Formal validation of industrial critical systems (Aix en Provence, France, October 1, 2013).
Jérôme Feret attended the workshop and gave a talk on the ASTRÉE analyzer.

CMSB: Computational Methods in Systems Biology (Klosterneuburg, Austria, September 23-25, 2013).
Ferdinanda Camporesi, Jérôme Feret, and Norman Ferns attended the conference. Jérôme Feret chaired a session and gave a talk on [12].

RAIM: 6ème Rencontres Arithmétiques de l'Informatique Mathématique (Paris, France, 18–20 November, 2013).
Antoine Miné attended the conference and gave a talk on generic and specific abstract domains for the static analysis of programs with floating-point arithmetic.

AVDCPS: Workshop on Analysis and Verification of Dependable Cyber Physical Software (Changsha, November 23-24, 2013).
Mehdi Bouaziz, Tie Cheng, Jérôme Feret, Xavier Rival, and Caterina Urban attended the conference. Jérôme Feret chaired a session. Mehdi Bouaziz, Tie Cheng, Jérôme Feret, Xavier Rival, and Caterina Urban gave talks.

9.1.5. Invitations and Participation in Seminars

— Arlen Cox gave a talk at LIAFA (Paris 7) about "QUIC Graphs: Relational Invariant Generation for Containers".

— Jérôme Feret gave a talk at IRIT (Toulouse, France) on the 28th of May, 2013 about model reduction in Kappa. He gave a talk on the OPENKAPPA platform at a meeting of the ANR BIOTEMPO project on the 30th of January, 2013 and a talk on context sensitive model reduction [12] at another meeting of the ANR BIOTEMPO project in Paris on the 10th of April, 2013. He gave a talk on context sensitive model reduction [12] at ETH Zürich on the 5th of June, 2013. He gave on talk on the analysis of digital filters [55], [56] at a meeting of the ANR VERASCO on the 16th of September, 2013.

— Norman Ferns gave two talks on the Kantorovich metric at the Computer Science Seminar of the University of Liège, Belgium, on the 5th of April 2013 and at the Technical Talk Seminar of the TU Dortmund University, Germany, on the 5th of July 2013.

— Antoine Miné gave a talk on "Constraint solving using numeric abstract domains" at the LIAFA Seminar "Vérification" (Paris, France) on April the 8th, 2013 and at the CEA LSL/LMeasi Seminar (Palaiseau, France) on June the 4th, 2013.

— Caterina Urban gave a talk on "The abstract domain of segmented ranking functions" at University of Udine (Italy) in March 2013.

— Xavier Rival gave a talk on the combination of memory abstractions at the IFIP 2.4 Working Group (Mysore, India, March 2013). He gave a talk on hierarchical shape abstract domains at IBM (Bangalore, India, March 2013). He gave a talk on the modular construction of shape analyzers at LIAFA (Université Paris 7, May 2013), at Cambridge University (Cambridge, UK, September 2013), at Seoul National University (Seoul, South Korea, October 2013), at KAIST (Daejon, South Korea, November 2013), at Hanyang University (Ansan, South Korea, November 2013), at the Institute of Software of the Chinese Accademy of Sciences (Beijing, November 2013). He gave a talk on the static analysis of safety critical embedded systems at Seoul National University (Seoul, South Korea, October 2013) and at KAIST (Daejon, South Korea, November 2013). He gave a talk on hierarchical shape abstract domains at Seoul National University (Seoul, South Korea, November 2013).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence :

- Medhi Bouaziz, Introduction to Programmation and Computer Sciences (Practical Works), 36ETD, L1, Université Paris-Diderot, France.
- Mehdi Bouaziz, Intensive course of algorithm, 35h ETD, Epita, Le Kremlin-Bicêtre, France.

- Mehdi Bouaziz, Preparation to the International Olympiad in Informatics, 40h ETD, Epita, Le Kremlin-Bicètre, France.
- Mehdi Bouaziz took the French team to the International Olympiad in Informatics at Brisbane, Australia.
- Tie Cheng, Introduction to algorithmics, 42h ETD, L3, École Polytechnique, Palaiseau, France.
- Jérôme Feret, and Caterina Urban, Mathematics, 40h ETD, L1, Licence Frontiers in Life Sciences (FdV), Université Paris-Descartes, France.
- Xavier Rival, Introduction to algorithmics, 40h ETD, L3, École Polytechnique, Palaiseau, France.
- Arnaud Spiwack, Introduction to recursive programming, 42h ETD, Université Pierre et Marie Curie (Paris 6), Paris, France.
- Antoine Toubans, Mathematics, 40hETD, L2, Université de Jussieu, Paris, France.

Master :

- Jérôme Feret, Computational Biology, 9h ETD, M1. Interdisciplinary Approaches to Life Science (AIV), Master Program, Université Paris-Descartes, France.
- Jérôme Feret, Antoine Miné, and Xavier Rival, Abstract Interpretation: application to verification and static analysis, 72h ETD, M2. Parisian Master of Research in Computer Science (MPRI). École normale supérieure. France.
- Xavier Rival, 20h ETD, M1, École Polytechnique, Palaiseau, France.
- Mehdi Bouaziz, Tie Cheng, Jérôme Feret, Antoine Miné, Xavier Rival, Caterina Urban, Abstract Interpretation, 30h ETD, M1-M2. East China Normal University, Shanghai, China.

Doctorat :

- Jérôme Feret, Abstract interpretation of intracellular signaling pathways, 4.5h ETD, CNRS Summer School on Formal modeling of Biological Regulatory Networks, Porquerolles, France.
- Antoine Miné, Inferring affine program invariants by abstract interpretation, 4.5h ETD, Spring School on Polyhedra Code Analysis and Optimization, Saint Germain au Mont d'Or, France, May 13–17, 2013.

9.2.2. Supervision

HdR : Antoine Miné, Static analysis by abstract interpretation of concurrent programs, École normale supérieure, 28 May 2013.

PhD in progress:

- Mehdi Bouaziz, Static analysis of security properties by abstract interpretation. November 2011, Patrick Cousot and Jérôme Feret, École Normale Supérieure.
- Ferdinanda Camporesi, Abstraction of Quantitative Semantics of Rule-based models, January 2009, Radhia Cousot and Jérôme Feret (co-directed thesis with Maurizio Gabrielli, University of Bologna).
- Arlen Cox, Representing dynamic languages heaps using parametric, modular, container abstract domains, 2013, Xavier Rival, École Normale Supérieure (co-directed thesis with Bor-Yuh Evan Chang, University of Colorado at Boulder).
- Tie Cheng, Static analysis of spreadsheet macros, October 2011, Xavier Rival, École Polytechnique.
- Vincent Laviron, Static Analysis of Functional Programs by Abstract Interpretation, October 2009, Patrick Cousot, École Normale Supérieure.

- Jiangchao Liu, Verification of the memory safety of a micro-kernel, December 2013, Xavier Rival, École Normale Supérieure
- Antoine Toubhans, Combination of shape abstract domains, October 2011, Xavier Rival, École Doctorale de Paris Centre
- Caterina Urban, Static Analysis of Functional Temporal Properties of Programs by Abstract Interpretation, November 2011, Radhia Cousot and Antoine Miné, École Normale Supérieure.

9.2.3. Juries

— Jérôme Feret reviewed the PhD thesis of Tatjana Petrov (ETH Zürich, Switzerland, JUne 6, 2013). He was also in the jury of Geoffroy Andrieux PhD thesis (Université Rennes 1, France, July 18, 2013).

— Antoine Miné reviewed the PhD thesis of Yassamine Seladji (CEA, Palaiseau, France, October 31, 2013). He was also in the jury of the PhD theses of Ramakrishna Upadrasta (Université Paris Sud, March 13, 2013) and Zhoulai Fu (Université Rennes 1, July 22, 2013).

— Xavier Rival reviewed the PhD theses of Valentin Perelle (Verimag, Université Joseph Fourrier, Grenoble, February 22, 2013) and of Gideon Smeding (Inria Rhône-Alpes, Université Joseph Fourrier, Grenoble, December 19, 2013).

9.3. Popularization

Xavier Rival gave an interview about the verification of safety critical embedded softwares to the “Inriality” online magazine, in July 2013.

10. Bibliography

Major publications by the team in recent years

[1] J. BERTRANE, P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, X. RIVAL. *Static Analysis and Verification of Aerospace Software by Abstract Interpretation*, in "Proceedings of the American Institute of Aeronautics and Astronautics (AIAA Infotech@Aerospace 2010)", Atlanta, Georgia, USA, American Institute of Aeronautics and Astronautics, 2010, http://hal.inria.fr/inria-00528611

[2] B. BLANCHET, P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, D. MONNIAUX, X. RIVAL. *A Static Analyzer for Large Safety-Critical Software*, in "Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)", ACM Press, June 7–14 2003, pp. 196–207

[3] P. COUSOT. *Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation*, in "Theoretical Computer Science", 2002, vol. 277, n^{o} 1–2, pp. 47–103

[4] J. FERET, V. DANOS, J. KRIVINE, R. HARMER, W. FONTANA. *Internal coarse-graining of molecular systems*, in "Proceeding of the national academy of sciences", Apr 2009, vol. 106, n^{o} 16, http://hal.inria.fr/inria-00528330

[5] L. MAUBORGNE, X. RIVAL. *Trace Partitioning in Abstract Interpretation Based Static Analyzers*, in "Proceedings of the 14th European Symposium on Programming (ESOP'05)", M. SAGIV (editor), Lecture Notes in Computer Science, Springer-Verlag, 2005, vol. 3444, pp. 5–20

[6] A. MINÉ. *The Octagon Abstract Domain*, in "Higher-Order and Symbolic Computation", 2006, vol. 19, pp. 31–100

[7] X. RIVAL. *Symbolic Transfer Functions-based Approaches to Certified Compilation*, in "Conference Record of the 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", ACM Press, New York, United States, 2004, pp. 1–13

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] A. MINÉ. , *Analyse statique par interprétation abstraite de programmes concurrents*, Ecole Normale Supérieure de Paris - ENS Paris, November 2013, Habilitation à Diriger des Recherches, http://hal.inria.fr/tel-00903447

Articles in International Peer-Reviewed Journals

[9] A. MINÉ. *Backward under-approximations in numeric abstract domains to automatically infer sufficient program conditions*, in "Science of Computer Programming", October 2013 [*DOI :* 10.1016/J.SCICO.2013.09.014], http://hal.inria.fr/hal-00903628

Invited Conferences

[10] P. COUSOT, R. COUSOT. *A galois connection calculus for abstract interpretation*, in "POPL - 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - 2014", San Diego, United States, S. JAGANNATHAN, P. SEWELL (editors), ACM, 2014, pp. 3-4 [*DOI :* 10.1145/2535838.2537850], http://hal.inria.fr/hal-00930103

International Conferences with Proceedings

[11] M. BARNETT, M. BOUAZIZ, F. LOGOZZO, M. FÄHNDRICH. *A case for static analyzers in the cloud*, in "Bytecode - 8th Workshop on Bytecode Semantics, Verification, Analysis, and Transformation", Rome, Italy, M. GOMEZ-ZAMALLOA, G. PUEBLA (editors), ENTCS, Elsevier, 2014, to appear, http://hal.inria.fr/hal-00925837

[12] F. CAMPORESI, J. FERET, J. HAYMAN. *Context-sensitive flow analyses: a hierarchy of model reductions*, in "CMSB - 11th Conference on Computational Methods in Systems Biology - 2013", Klosterneuburg, Austria, A. GUPTA, T. A. HENZINGER (editors), Lecture Notes in BioInformatics, Springer, 2013, vol. 8130, pp. 220-233 [*DOI :* 10.1007/978-3-642-40708-6_17], http://hal.inria.fr/hal-00846893

[13] B.-Y. E. CHANG, X. RIVAL. *Modular Construction of Shape-Numeric Analyzers*, in "Festschrift for Dave Schmidt", Manhattan, Kansas, United States, A. BANERJEE, O. DANVY, K.-G. DOH, J. HATCLIFF (editors), EPTCS, September 2013, vol. 129, http://hal.inria.fr/hal-00926948

[14] P. COUSOT, R. COUSOT, M. FÄHNDRICH, F. LOGOZZO. *Automatic Inference of Necessary Preconditions*, in "VMCAI 2013 - 14th Conference on Verification, Model Checking and Abstract Interpretation", Rome, Italy, R. GIACOBAZZI, J. BERDINE, I. MASTROENI (editors), LNCS - Lecture Notes in Computer Science, Springer, 2013, vol. 7737, pp. 128-148 [*DOI :* 10.1007/978-3-642-35873-9_10], http://hal.inria.fr/hal-00930070

[15] J. HAYMAN. *Filtered compression for Kappa*, in "SASB - 4th INternational Workshop on Static Analysis and Systems Biology", Seattle, United States, J. FERET, A. LEVCHENKO (editors), Elsevier, 2014, to appear, http://hal.inria.fr/hal-00925549

[16] J. HAYMAN, T. HEINDEL. *Pattern Graphs and Rule-Based Models: The Semantics of Kappa*, in "FOSSACS - 16th International Conference on Foundations of Software Science and Computation Structures", Rome, Italy, F. PFENNING (editor), Lecture Notes in Computer Science, Springer, 2013, vol. 7794, pp. 1–16 [*DOI :* 10.1007/978-3-642-37075-5], http://hal.inria.fr/hal-00925345

[17] A. MINÉ. *Relational thread-modular static value analysis by abstract interpretation*, in "VMCAI 2014 - 15th International Conference on Verification, Model Checking, and Abstract Interpretation", San Diego, United States, K. MCMILLAN, X. RIVAL (editors), Lecture Notes in Computer Science, Springer, January 2014, vol. 8318, pp. 39-58 [*DOI :* 10.1007/978-3-642-54013-4_3], http://hal.inria.fr/hal-00925713

[18] M. PELLEAU, A. MINÉ, C. TRUCHET, F. BENHAMOU. *A Constraint Solver based on Abstract Domains*, in "VMCAI 2013 - 14th International Conference on Verification, Model Checking, and Abstract Interpretation", Rome, Italy, R. GIACOBAZZI, J. BERDINE, I. MASTROENI (editors), Lecture Notes in Computer Science, Springer-Verlag, 2013, vol. 7737, pp. 434–454 [*DOI :* 10.1007/978-3-642-35873-9_26], http://hal.inria.fr/hal-00785604

[19] M. PELLEAU, A. MINÉ, C. TRUCHET, F. BENHAMOU. *Un solveur de contraintes basé sur les domaines abstraits*, in "9èmes Journées Francophones de Programmation par Contraintes", Aix-en-Provence, France, June 2013, pp. 259-268, http://hal.inria.fr/hal-00925430

[20] A. TOUBHANS, B.-Y. E. CHANG, X. RIVAL. *Reduced Product Combination of Abstract Domains for Shapes*, in "VMCAI 2013 : 14th International Conference on Verification, Model Checking and Abstract Interpretation", Rome, Italy, R. GIACOBAZZI, J. BERDINE, I. MASTROENI (editors), Lecture Notes in Computer Science, Springer, January 2013, vol. 7737, pp. 375-395 [*DOI :* 10.1007/978-3-642-35873-9_23], http://hal.inria.fr/hal-00760428

[21] O. TRIPP, M. PISTOIA, P. COUSOT, R. COUSOT, S. GUARNIERI. *Andromeda: Accurate and Scalable Security Analysis of Web Applications*, in "FASE 2013 - International Conference Fundamental Approaches to Software Engineering", Rome, Italy, V. CORTELLESSA, D. VARRÓ (editors), LNCS - Lecture Notes in Computer Science, Springer, 2013, vol. 7793, pp. 210-225, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013 [*DOI :* 10.1007/978-3-642-37057-1_15], http://hal.inria.fr/hal-00930096

[22] C. URBAN, A. MINÉ. *An abstract domain to infer ordinal-valued ranking functions*, in "23rd European Symposium on Programming", Grenoble, France, Z. SHAO (editor), Lecture Notes in Computer Science, Springer, April 2014, http://hal.inria.fr/hal-00925731

[23] C. URBAN. *Piecewise-Defined Ranking Functions*, in "13th International Workshop on Termination", Bertinoro, Italy, J. WALDMANN (editor), August 2013, pp. 69-73, http://hal.inria.fr/hal-00925682

[24] C. URBAN. *The Abstract Domain of Segmented Ranking Functions*, in "Static Analysis, 20th International Symposium,", Seattle, United States, F. LOGOZZO, M. FÄHNDRICH (editors), Lecture Notes in Computer Science, Springer, June 2013, vol. 7935, pp. 43-62 [*DOI :* 10.1007/978-3-642-38856-9_5], http://hal.inria.fr/hal-00925670

Books or Proceedings Editing

[25] R. GIACOBAZZI, R. COUSOT (editors). , *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13*, ACM, 2013, 574 p. , http://hal.inria.fr/hal-00930257

[26] K. MCMILLAN, X. RIVAL (editors). , *Verification, Model Checking, and Abstract Interpretation*, Springer, 2014, vol. 8318, 493 p. , http://hal.inria.fr/hal-00931694

Other Publications

[27] A. ATKI. , *Implémentation du domaine asbtrait numérique TVPI pour APRON*, École PolytechniquePalaiseau, July 2013, 28 p. , http://hal.inria.fr/hal-00925999

[28] M. BRY. , *Synchronisation de fils d'exécution en interprétation abstraite*, École PolytechniquePalaiseau, July 2013, 16 p. , http://hal.inria.fr/hal-00926979

[29] C. URBAN. , *Automatic Inference of Ranking Functions by Abstract Interpretation*, January 2014, Student Poster Session, 41st Symposium on Principles of Programming Languages (POPL 2014), http://hal.inria.fr/hal-00925760

References in notes

[30] P. COUSOT. *Proving the Absence of Run-Time Errors in Safety-Critical Avionics Code, invited tutorial*, in "Proceedings of the Seventh ACM & IEEE International Conference on Embedded Software, EMSOFT'2007", C. M. KIRSCH, R. WILHELM (editors), ACM Press, New York, USA, 2007, pp. 7–9

[31] P. COUSOT. , *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes (in French)*, Université scientifique et médicale de Grenoble, 21 March 1978

[32] R. COUSOT. , *Reasoning about program invariance proof methods*, Centre de Recherche en Informatique de Nancy (CRIN), Institut National Polytechnique de LorraineNancy, France, July 1980, n^{o} CRIN-80-P050, 22 p.

[33] P. COUSOT. *Semantic Foundations of Program Analysis*, in "Program Flow Analysis: Theory and Applications", S. S. MUCHNICK, N. D. JONES (editors), Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981, chap. 10, pp. 303–342

[34] R. COUSOT. , *Proving invariance properties of parallel programs by backward induction*, University Paul VerlaineMetz, France, March 1981, n^{o} LRIM-82-02, 38 p.

[35] R. COUSOT. , *Fondements des méthodes de preuve d'invariance et de fatalité de programmes parallèles (in French)*, Institut National Polytechnique de LorraineNancy, France, 21 November 1985

[36] P. COUSOT. *The Calculational Design of a Generic Abstract Interpreter, invited chapter*, in "Calculational System Design", M. BROY, R. STEINBRÜGGEN (editors), NATO Science Series, Series F: Computer and Systems Sciences. IOS Press, Amsterdam, The Netherlands, 1999, vol. 173, pp. 421–505

[37] P. COUSOT, R. COUSOT. *Basic Concepts of Abstract Interpretation, invited chapter*, in "Building the Information Society", R. JACQUART (editor), Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004, chap. 4, pp. 359–366

[38] P. COUSOT, R. COUSOT. *Grammar Analysis and Parsing by Abstract Interpretation, invited chapter*, in "Program Analysis and Compilation, Theory and Practice: Essays dedicated to Reinhard Wilhelm on the Occasion of his 60th Birthday", T. W. REPS, M. SAGIV, J. BAUER (editors), Lecture Notes in Computer Science, Springer, Berlin, Germany, 2007, vol. 4444

[39] P. COUSOT, R. COUSOT. *Bi-inductive structural semantics*, in "Information and Computation", 2009, vol. 207, n^{o} 2, pp. 258–283

[40] P. COUSOT, R. COUSOT. *An Abstract Interpretation Framework for Termination*, in "Conference Record of the 39 th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", Philadelphia, PA, ACM Press, New York, January 25-27 2012, pp. 245–258

[41] P. COUSOT, R. COUSOT. *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in "Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", ACM Press, New York, United States, 1977, pp. 238–252

[42] P. COUSOT, R. COUSOT. *Systematic design of program analysis frameworks*, in "Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", San Antonio, Texas, ACM Press, New York, NY, USA, 1979, pp. 269–282

[43] P. COUSOT, R. COUSOT. *Semantic analysis of communicating sequential processes*, in "Seventh International Colloquium on Automata, Languages and Programming", J. W. DE BAKKER, J. VAN LEEUWEN (editors), Lecture Notes in Computer Science 85, Springer-Verlag, Berlin, Germany, July 1980, pp. 119–133

[44] P. COUSOT, R. COUSOT. *Invariance Proof Methods and Analysis Techniques For Parallel Programs*, in "Automatic Program Construction Techniques", A. W. BIERMANN, G. GUIHO, Y. KODRATOFF (editors), Macmillan, New York, New York, United States, 1984, chap. 12, pp. 243–271

[45] P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, D. MONNIAUX, X. RIVAL. *The* ASTRÉE *analyser*, in "Proceedings of the Fourteenth European Symposium on Programming Languages and Systems, ESOP'2005, Edinburg, Scotland", M. SAGIV (editor), Lecture Notes in Computer Science, Springer, Berlin, Germany, 2–10 April 2005, vol. 3444, pp. 21–30

[46] P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, D. MONNIAUX, X. RIVAL. *Varieties of Static Analyzers: A Comparison with* ASTRÉE*, invited paper*, in "Proceedings of the First IEEE & IFIP International Symposium on Theoretical Aspects of Software Engineering, TASE'07", Shanghai, China, M. HINCHEY, J. HE, J. SANDERS (editors), IEEE Computer Society Press, Los Alamitos, California, USA, 6–8 June 2007

[47] P. COUSOT, R. COUSOT, R. GIACOBAZZI. *Abstract Interpretation of Resolution-Based Semantics*, in "Theoretical Computer Science", Nov. 2009, vol. 410, n^{o} 46

[48] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. HAYMAN, J. KRIVINE, C. THOMPSON-WALSH, G. WINSKEL. *Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models*, in "32nd IARCS Annual

Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)", Hyderabad, India, D. D'Souza, J. Radhakrishnan, K. Telikepalli (editors), LIPIcs, December 2012, http://hal.inria.fr/hal-00734487

[49] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine. *Abstracting the differential semantics of rule-based models: exact and automated model reduction*, in "Proceedings of Logic in Computer Science (LICS 2010), Edinburgh, UK", J.-P. Jouannaud (editor), 2010, pp. 362–381, http://hal.inria.fr/hal-00520112, http://hal.inria.fr/hal-00520112

[50] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine. *Rule-based modelling of cellular signalling*, in "Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)", Portugal, September 2007, vol. 4703, pp. 17–41, http://hal.archives-ouvertes.fr/hal-00164297/en/

[51] V. Danos, J. Feret, W. Fontana, J. Krivine. *Scalable Simulation of Cellular Signaling Networks*, in "Proceedings of the 5th Asian Symposium on Programming Languages and Systems - APLAS'07", Z. Shao (editor), Lecture Notes in Computer Science, Springer, 2007, vol. 4807, pp. 139-157 [*DOI :* 10.1.1.139.5120], http://hal.inria.fr/inria-00528409/en/

[52] V. Danos, J. Feret, W. Fontana, J. Krivine. *Abstract Interpretation of Cellular Signalling Networks*, in "Proceedings of the 9th International Conference on Verification, Model Checking and Abstract Interpretation - VMCAI'08", F. Logozzo, D. A. Peled, L. D. Zuck (editors), Lecture Notes in Computer Science, Springer, 2008, vol. 4905, pp. 83-97 [*DOI :* 10.1007/978-3-540-78163-9_11], http://hal.inria.fr/inria-00528352/en/

[53] V. Danos, C. Laneve. *Formal Molecular Biology*, in "Theoretical Computer Science", 10 2004, vol. 325, n^{o} 1, pp. 69-110 [*DOI :* 10.1016/j.tcs.2004.03.065], http://hal.archives-ouvertes.fr/hal-00164591/en/

[54] J. Desharnais, R. Jagadeesan, V. Gupta, P. Panangaden. *The Metric Analogue of Weak Bisimulation for Probabilistic Processes*, in "LICS", IEEE Computer Society, 2002, pp. 413-422

[55] J. Feret. *Static Analysis of Digital Filters*, in "European Symposium on Programming (ESOP'04)", LNCS, Springer-Verlag, 2004, n^{o} 2986, pp. 33–48, © Springer-Verlag

[56] J. Feret. *Numerical Abstract Domains for Digital Filters*, in "International workshop on Numerical and Symbolic Abstract Domains (NSAD 2005)", 2005

[57] N. Ferns, P. Panangaden, D. Precup. *Bisimulation Metrics for Continuous Markov Decision Processes*, in "SIAM J. Comput.", 2011, vol. 40, n^{o} 6, pp. 1662-1714

[58] R. Harmer, V. Danos, J. Feret, J. Krivine, W. Fontana. *Intrinsic Information carriers in combinatorial dynamical systems*, in "Chaos", 2010, vol. 20, n^{o} 3, http://hal.inria.fr/hal-00520128, http://hal.inria.fr/hal-00520128

[59] A. Müller. , *Integral probability metrics and their generating classes of functions*, Discussion paper, Inst. für Wirtschaftstheorie und Operations Research, 1995, http://books.google.fr/books?id=ZYNYtwAACAAJ

[60] F. van Breugel, J. Worrell. *An Algorithm for Quantitative Verification of Probabilistic Transition Systems*, in "CONCUR", K. G. Larsen, M. Nielsen (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2154, pp. 336-350

[61] F. VAN BREUGEL, J. WORRELL. *Towards Quantitative Verification of Probabilistic Transition Systems*, in "ICALP", F. OREJAS, P. G. SPIRAKIS, J. VAN LEEUWEN (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2076, pp. 421-432