IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2011

Project-Team ABSTRACTION

Abstract Interpretation and Static Analysis

IN COLLABORATION WITH: Laboratoire d'Informatique de l'Ecole Normale Supérieure (LIENS)

RESEARCH CENTER
Paris - Rocquencourt

THEME
Programs, Verification and Proofs

Table of contents

Project-Team ABSTRACTION

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

ABSTRACTION *is located at the École normale supérieure, Paris.*

1. Members

Research Scientists
Radhia Cousot [Senior Researcher, CNRS, HdR]
Jérôme Feret [Junior Researcher, INRIA Paris–Rocquencourt]
Antoine Miné [Junior Researcher, CNRS]
Xavier Rival [Junior Researcher, INRIA Paris–Rocquencourt, HdR]

Faculty Members
Julien Bertrane [— Aug. 2011]
Patrick Cousot [Team leader, Professor, ENS, HdR]

PhD Students
Mehdi Bouaziz [Nov. 2011 —]
Ferdinanda Camporesi
Vincent Laviron
Cheng Tie [Oct. 2011 —]
Antoine Toubhans [Sep. 2011 —]
Caterina Urban [Dec. 2011 —]
Matteo Zanioli

Post-Doctoral Fellows
Jonathan Hayman [Nov. 2011 —]
Tahina Ramananandro [Sep. 2011 —]
Alessandro Romanel [Jan. 2011 — Nov. 2011]
Pascal Sotin [Oct. 2011 —]

Visiting Scientists
David Delmas [Sep. 2011 —]
Yanjun Wen [Jun. 2011 —]

Administrative Assistants
Joëlle Isnard [Administrative Head DI, ENS]
Marine Meyer [INRIA, Apr. 2011 —]

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

The paper "Static Analysis and Verification of Aerospace Software by Abstract Interpretation", written by the team [1], has been selected in 2011 by the AIAA Intelligent Systems Technical Committee as the Best Paper from the AIAA 2010 Infotech@Aerospace Conference.

The MemCAD ERC Starting Grant ("Memory Compositional Abstract Domains") was started on October, 1st. 2011 (funded by the European Research Counsil "IDEAS" programme).

3. Scientific Foundations

3.1. Abstract Interpretation Theory

The abstract interpretation theory [37], [28], [38], 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 [37], [30], or parallel [32] with shared memory [29], [31], [40] or synchronous message [39] 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. [35], [36], [44]) 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 [42]. 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 [37], [38];
- guiding the correct formal design of *abstract semantics* [36], [44] and automatic tools for *program analysis* (computing an abstract semantics) and *program verification* (proving that an abstract semantics satisfies an abstract specification) [33].

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 recent, short, informal, and intuitive introduction to the theory of abstract interpretation can be found in [33], 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" [34] 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

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 [27], [43], 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

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

5.1. The Apron Numerical Abstract Domain Library

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

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).

5.2. The Astrée Static Analyzer of Synchronous Software

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

ASTRÉE is a static analyzer for sequential programs based on abstract interpretation [37], [28], [38], [30].

The ASTRÉE static analyzer [27], [43][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 scientifique leader, correspondent], Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival.

ASTRÉEA is a static analyzer prototype for parallel software based on abstract interpretation [39], [40], [32]. 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 29h on a 2.66 GHz 64-bit intel server using one core and generates around 1,800 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. Improvements were made this year concerning the precision of ASTRÉEA (from 7,600 alarms in 2010 to 1,800 now) and will continue within the scope of the ASTRÉEA ANR project (Section 8.1.1.2).

The details of the analysis are described in [22].

5.4. OpenKappa

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

OPENKAPPA is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language [49], a static analyzer [48] (for debugging models), a simulator [47], a compression tool for causal traces [46], and a model reduction tool [4], [45], [50].

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.

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 Ecole Polytechnique Federale de Lausanne, 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).

5.5. Translation Validation

Participant: Xavier Rival [correspondent].

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 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.

5.6. Zarith

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

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. Abstractions of Functions

Participants: Patrick Cousot, Radhia Cousot.

The idea of domain segmentation for arrays [18] has been extended to the abstraction of functions [41] by combination of a partionning of their domain of definition and a functional or relational abstraction of blocks into their co-domain [17].

6.2. Analysis of Biological Pathways

We have improved our framework to design and analyze biological networks. This framework focused 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).

6.2.1. Automatic Reduction of Differential Semantics

Participants: Ferdinanda Camporesi, Vincent Danos [University of Edinburgh], Jérôme Feret, Walter Fontana [Harvard Medical School], Russ Harmer [Harvard Medical School], Jean Krivine [Paris VII].

We have developed an abstract interpretation-based framework that enables the reduction of the differential semantics for protein-protein interaction networks. Results are sound since trajectories in the abstract system are projections of the trajectories in the concrete system.

The flow of information is a key element in our model reduction framework because it enables the identification of the correlations which are useless when computing observables of interest. Thus there is a need of providing good trade-off in the description of the flow of information throughout the biochemical structure of chemical species.

The notion of symmetries between sites is also important, since knowing that two sites have exactly the same capabilities of interaction enable exact quotienting (or lumping) of the set of reachable species.

In [13], [14], we have proposed a heterogeneous over-approximation of the flow of information where the flow that is attached to an agent can depend on its relative position in a chemical species. Moreover, we have showed how to use symmetries between sites so as to define another model reduction and we have proposed an algebraic product to combine model reductions, the product of two reduced models being the least abstract model which is at least as abstract as both model.

6.2.2. Automatic Reduction of Stochastic Semantics

Participants: Ferdinanda Camporesi, Jérôme Feret, Thomas Henzinger [Institute of Science and Technology, Austria], Heinz Koeppl [ETH Zürich], Tatjana Petrov [ETH Zürich].

We have proposed an abstract interpretation-based framework for reducing the state-space of stochastic semantics for protein-protein interaction networks. Our framework ensures that the trace distribution of the reduced system is the exact projection of the trace distribution of the concrete system. Moreover, when the abstraction is complete, if any pair of concrete states that have the same abstraction are equipropable at initial state, any pair of concrete states that share the same abstraction are equiprobable at any time t.

In [12], we have formalized the model reduction framework for the stochastic semantics and we have established the relationships with the notions of lumpability, and bisimulation is established.

6.3. Automatic Array Content Analysis by Segmentation

Participants: Patrick Cousot, Radhia Cousot, Francesco Logozzo [Microsoft Research (Redmond, USA)].

In [18], we introduce `FunArray`, a parametric segmentation abstract domain functor for the fully automatic and scalable analysis of array content properties. The functor enables a natural, painless and efficient lifting of existing abstract domains for scalar variables to the analysis of uniform compound data-structures such as arrays and collections (as well as matrices when instantiating the functor on itself). The analysis automatically and semantically divides arrays into consecutive non-overlapping possibly empty segments. Segments are delimited by sets of bound symbolic expressions and abstracted uniformly. All bound expressions appearing in a set are equal in the concrete. The `FunArray` can be naturally combined via reduced product with any existing analysis for scalar variables. The bound expressions, the segment abstractions and the reduction operator are the three parameters of the analysis. Once the functor has been instantiated with fixed parameters, the analysis is fully automatic.

We first prototyped `FunArray` in `Arrayal` to adjust and experiment with the abstractions and the algorithms to obtain the appropriate precision/ratio cost. Then it was implemented into CCCHECK (formerly CLOUSOT), an abstract interpretation-based static contract checker for `.NET` by Francesco Logozzo. The precision and the performance of the analysis has been empirically validated by running it on the main libraries of `.NET` and on its own code. It was able to infer thousands of invariants and to verify the implementation with a modest overhead (circa 1%). To the best of our knowledge this is the first analysis of this kind applied to such a large code base, and proven to scale.

6.4. Extrapolation operators for combinations of abstract domains

Participants: Agostino Cortesi [Università Ca'Foscardi di Venizia], Matteo Zanioli.

Extrapolation operators are crucial to ensure the scalability of the analysis to large software systems. In [10], we set the ground for a systematic design of widening and narrowing operators, by comparing the different definitions introduced in the literature and by discussing how to tune them in case of domain abstraction and domains' combination through Cartesian and reduced products.

6.5. Grammar Semantics, Analysis and Parsing

Participants: Patrick Cousot, Radhia Cousot.

In [11], we study the abstract interpretations of a fixpoint protoderivation semantics defining the maximal derivations of a transitional semantics of context-free grammars akin to pushdown automata. The result is a hierarchy of bottom-up or top-down semantics refining the classical equational and derivational language semantics and including Knuth grammar problems, classical grammar flow analysis algorithms, and parsing algorithms.

6.6. Information Flow

The analysis of the flow of information in a program consists in detecting the propagation of sensitive information through the program points of this program thanks to a dependency analysis.

6.6.1. Dependency Analysis and Numerical Invariants

Participants: Agostino Cortesi [Università Ca'Foscardi di Venizia], Matteo Zanioli.

A new framework has been proposed in [16], that combines variable dependency analysis, based on propositional formulas, and variables' value analysis, based on generic numerical domains.

6.6.2. Leakage Analysis

Participants: Matteo Zanioli [Correspondent], Pietro Ferrara [ETH, Zurich], Agostino Cortesi [Università Ca' Foscari].

In [24], we present SAILS, a new tool that combines SAMPLE, a generic static analyzer, and a sophisticated domain for leakage analysis. This tool does not require to modify the original language, since it works with mainstream languages like JAVA™, and it does not require any manual annotation. SAILS can combine the information leakage analysis with different heap abstractions, inferring information leakage over programs with complex data structures. SAILS has been applied to the analysis of the SecuriBench-micro suite. The experimental results underline the effectiveness of the analysis, since SAILS is in position to analyze several benchmarks in about 1 second without producing false alarms in more than 90% of the programs.

6.7. Linear Absolute Value Relation Analysis

Participants: Liqian Chen [National Laboratory for Parallel and Distributed Processing, Changsha, P. R. China], Antoine Miné, Ji Wang [National Laboratory for Parallel and Distributed Processing, Changsha, P. R. China], Patrick Cousot.

We present in [15] an abstract domain dealing with linear inequalities involving variables together with their absolute values. It is an extension of the classical linear relation analysis, which permits to deal with some non convex numerical sets. A first nice result states the equivalence between these "linear absolute value inequalities" (AVI) and "interval linear inequalities", and "extended linear complementary inequalities" (XLCP, pairs of positive solutions whose pairwise components are not both not zero). The key contribution is the extension of the double-description of polyhedra to XLCP solutions, which is then used to define the standard operations on AVI. The method has been implemented, and experiments show interesting results, with reasonable performances with respect to linear relation analysis.

6.8. Probabilistic Analysis

Participants: Patrick Cousot, Michaël Monerau.

The abstract interpretation theory has been widely used in the past decades for verifying properties of computer systems. We have introduced a new extension of this well-known framework to the case of probabilistic systems [21].

The probabilistic abstraction framework we propose allows to systematically lift any classical analysis or verification method to the probabilistic setting by separating in the program semantics the probabilistic behavior from the (non-)deterministic behavior. This separation provides new insights for designing novel probabilistic static analyses and verification methods.

We have defined concrete probabilistic semantics and proposed different ways to abstract them. The approach is expressive and effective. The previous techniques for probabilistic analysis are actually abstractions expressible in our framework.

6.9. Safety

Participants: Patrick Cousot, Radhia Cousot.

The abstract interpretation design principle has been applied to the design of new forward and backward proof, verification and analysis methods for safety [17]. The safety collecting semantics defining the strongest safety property of programs is first expressed in a constructive fixpoint form. Safety proof and checking/verification methods then immediately follow by fixpoint induction. Static analysis of abstract safety properties such as invariance are constructively designed by fixpoint abstraction (or approximation) to (automatically) infer safety properties.

6.10. Security

Participants: Patrick Cousot, Radhia Cousot.

We have developed, episodically since 2007, an abstract interpretation framework for security and program securization that is the transformation of a program into a secured program satisfying security criteria defined by a human or artificial supervisor (this is verification when no transformation is needed). The securization is based on the notion of responsibility analysis determining which choices in the program (inputs, random draws, interrupts, schedules, etc.) can definitely cause or avoid desired or menacing events, or have no control at all on the occurrence of these events. Various securization policies (eager, early or late lazy, etc.) have been identified to prevent or enforce the occurrence of events.

6.11. Shape Analysis

We have extended the XISA (eXtensible Inductive Shape Analysis) framework, in order to better deal with low level coding styles and programming languages, and in order to analyze recursive programs in a context dependent way. We also introduced a classification for semantic memory models.

6.11.1. Abstracting Calling-Context with Shapes

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

Interprocedural program analysis is often performed by computing procedure summaries. While possible, computing adequate summaries is difficult, particularly in the presence of recursive procedures. In [23], we propose a complementary framework for interprocedural analysis based on a direct abstraction of the calling context. Specifically, our approach exploits the inductive structure of a calling context by treating it directly as a stack of activation records. We built an abstraction based on separation logic with inductive definitions. A key element of this abstract domain is the use of parameters to refine the meaning of such call stack summaries and thus express relations across activation records and with the heap. In essence, we define an abstract interpretation-based analysis framework for recursive programs that permits a fluid per call site abstraction of the call stack—much like how shape analyzers enable a fluid per program point abstraction of the heap.

6.11.2. Abstract domains for the analysis of programs manipulating complex data-structures

Participant: Xavier Rival.

We proposed a framework for building abstract domains for the static analysis of programs which manipulate complex* data-structures [8]. Our abstract domain is parametric in the choice of a numerical abstract domain to represent properties of numeric memory cells and in the choice of a set of inductive definitions to be used in order to summarize unbounded heap regions. It features standard primitives for the computation of transfer functions, for the inclusion checking and for the computation of widening iterates. We also proposed an extension to handle programs that make use of low-level memory addressing, and proposed an extension of the widening to infer inductive definitions.

6.11.3. Composite abstract domain for the analysis of dynamic structures

Participants: Xavier Rival, Antoine Toubhans.

Reduced product is a general operation to combine abstract domains into more powerful abstract domains, which has been especially used to construct numerical abstract domains. However, until now, it has not been applied to memory structures. We proposed an instance of a reduced product operation, which can be applied on shape abstract domains based on separation logic and on inductive definitions. The advantage of this construction is that it allows to describe more complex heap dynamic data structures without making the design of all abstract operation more complex. In the other hand, it incurs a reduction cost, whenever we need to transport some information from one domain to the other. We showed that optimal reduction cannot be achieved, and identified the main source of complexity of this operation. A prototype implementation was also carried out. This work was done as part of Antoine Toubhans Master internship.

6.12. Static Analysis of Parallel Software

Participant: Antoine Miné.

We present in [22] a static analysis by abstract interpretation to check for run-time errors in parallel C programs. Following our work on ASTRÉE, we focus on embedded critical programs without recursion nor dynamic memory allocation, but extend the analysis to a static set of threads. Our method iterates a slightly modified non-parallel analysis over each thread in turn, until thread interferences stabilize. We prove the soundness of the method with respect to a sequential consistent semantics and a reasonable weakly consistent memory semantics. We then show how to take into account mutual exclusion and thread priorities through partitioning over the scheduler state. We present preliminary experimental results analyzing a real program with our prototype ASTRÉEA (see 5.3) and demonstrate the scalability of our approach.

6.13. Termination

Participants: Patrick Cousot, Radhia Cousot.

In [17], we have introduced an abstract interpretation for termination. Proof, verification and analysis methods for termination all rely on two induction principles: (1) a variant function or induction on data ensuring progress towards the end and (2) some form of induction on the program structure.

So far, no clear design principle did exist for termination as is the case for safety so that the existing approaches are scattered and largely not comparable with each other.

For (1), we show that this design principle applies equally well to potential and definite termination. The trace-based termination collecting semantics is given a fixpoint definition. Its abstraction yields a fixpoint definition of the best variant function. By further abstraction of this best variant function, we derive the Floyd/Turing termination proof method as well as new static analysis methods to effectively compute approximations of this best variant function.

For (2), we introduce a generalization of the syntactic notion of structural induction (as found in Hoare logic) into a semantic structural induction based on the new semantic concept of inductive trace cover covering execution traces by segments, a new basis for formulating program properties. Its abstractions allow for generalized recursive proof, verification and static analysis methods by induction on both program structure, control, and data. Examples of particular instances include Floyd's handling of loop cut-points as well as nested loops, Burstall's intermittent assertion total correctness proof method, and Podelski-Rybalchenko transition invariants.

6.14. Theories, Solvers and Static Analysis

Participants: Patrick Cousot, Radhia Cousot, Laurent Mauborgne [IMDEA Software (Madrid, Spain)].

In [20], we have introduced a reduced product combining algebraic and logical abstractions to design program correctness verifiers and static analyzers by abstract interpretation. The key new idea is to show that the Nelson-Oppen procedure for combining theories in SMT-solvers computes a reduced product in an observational semantics, so that algebraic and logical abstract interpretations can naturally be combined in a classical way using a reduced product on this observational semantics. The main practical benefit is that reductions can be performed within the logical abstract domains, within the algebraic abstract domains, and also between the logical and the algebraic abstract domains, including the case of abstractions evolving during the analysis.

6.15. Underapproximation for Precondition Inference

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

In the context of program design by contracts, programmers often insert assertions in their code to be optionally checked at runtime, at least during the debugging phase. These assertions would better be given as a precondition of the method/procedure in which they appear. Potential errors would be discovered earlier and, more importantly, the precondition could be used in the context of separate static program analysis as part of the abstract semantics of the code. However in the case of collections (data structures such as arrays, lists, etc) checking both the precondition and the assertions at runtime appears superfluous and costly. So the precondition is often omitted since it is checked anyway at runtime by the assertions. It follows that the static analysis can be much less precise, a fact that can be difficult to understand since "the precondition and assertions are equivalent" (i.e. at runtime, up to the time at which warnings are produced, but not statically) e.g. for separate static analysis. Moreover preconditions are often understood as overapproximations and thus may exclude good runs which is counter-intuitive for programmers. On the contrary, with considering underapproximations [37], [28] which exclude no good run, ensures that if the precondition is violated then a runtime error must definitely be raised later, and if the precondition is not strong enough to catch all errors they will definitely be captures by a later runtime check.

In [19], we define precisely and formally the contract inference problem from intermittent assertions on scalar variables and elements of collections inserted in the code by the programmer. Our definition excludes no good run even when a non-deterministic choice (e.g. an interactive input) could lead to a bad one. We then introduce new abstract interpretation-based methods to automatically infer both the static contract precondition of a method/procedure and the code to check it at runtime on scalar and collection variables. It has been implemented in CCCHECK (formerly CLOUSOT) by Francesco Logozzo and Manuel Fähndrich.

6.16. Verification of spreadsheet programs by abstract interpretation

Participants: Tie Cheng, Xavier Rival.

Spreadsheet tools (Excel, Openoffice) come with powerful languages which can manipulate sheets in various ways. However, no type discipline is enforced, so that the programs may corrupt spreadsheet contents in many ways. We proposed an abstraction to describe sets of valid spreadsheet states, and designed a verifier for invariants expressed in this abstract domain. Our verifier assumes invariants are defined at the head of loops in the programs (as widening operators for the inference of loop invariants). This work was done as part of Tie Cheng Master internship.

7. Contracts and Grants with Industry

7.1. Contracts with Industry

7.1.1. Contracts

7.1.1.1. Anastasy

Title: ANASTASY

Type: Industrial contract

Duration: September 2009 - December 2011

Others partners: Airbus France

Abstract: ANASTASY (ANAlyse STAtique aSYnchone) is an industrial project with Airbus France on the static program analysis of asynchronous programs by abstract interpretation which objective is determined annually. Patrick Cousot is the principal investigator for this action.

7.1.2. License agreement

7.1.2.1. 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.

7.2. Grants with Industry

7.2.1. FNRAE projects

7.2.1.1. Ascert

Title: Analyses Statiques CERTifiés

Type: 6th call: Verification methods for software and systems

Instrument: FNRAE grant

Duration: April 2009 - March 2012

Coordinator: INRIA (France)

Others partners: INRIA-Bretagne Atlantique, the INRIA Rhône-Alpes, the INRIA Paris-Rocquencourt, and the ENS.

See also: http://ascert.gforge.inria.fr/

Abstract: Although static analyzers have demonstrated their ability to prove the absence of large classes of errors in critical software, they are themselves large and complex software, so it is natural to question their implementation correctness and the validity of their output. The focus of the ASCERT project is the use of formal methods to ensure the correctness of an analyzer with respect to the abstraction interpretation theory. Methods to be investigated include the direct proof of the analyzer, the proof of a verifier for the analyzer result, and the validation of the inductive invariants generated by the analyzer, using the Coq proof assistant. These methods will be applied to the certification of several numerical abstract domains, of an abstract interpreter for imperative programs and its possible extensions to one of the formal semantics of the CompCert verified C compiler.

7.2.1.2. Sardanes

Title: Sémantique, Analyse et tRansformation Des Applications Numériques Embarqués Synchrones

Type: 6th call: Verification methods for software and systems

Instrument: FNRAE grant

Duration: February 2009 - September 2013

Coordinator: Université de Perpignan

Others partners: Université de Perpignan and the ENS.

See also: http://perso.univ-perp.fr/mmartel/sardanes.html

Abstract: SCADE is widely used to write critical embedded software, as a specification and verification language. The semantics of SCADE uses real arithmetics whereas it is compiled into a language that uses floating-point arithmetics. The goal of the SARDANES project is to use expression transformation so as to ensure that the numerical properties of the programs is preserved during the compilation. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR projects

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. 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. EU Project

8.2.1.1. MBAT

Title: Combined Model-based Analysis & Testing of Embedded Systems

Type: Artemis Call 10

Instrument: FP7 project

Duration: November 2011 - October 2014

Coordinator: Daimler (Germany)

Others partners: 38 partners in Austria, Denmark, Estonia, France, Germany, Italy, Sweden, and United Kingdom

See also: http://www.artemis-ia.eu/project/index/view/?project=29

Abstract: MBAT will mainly focus on providing a technology platform for effective and cost-reducing validation and verification of embedded systems, focusing primarily on transportation domain, but also to be used in further domains. The project involves thirty three European industrial (large companies and SMEs) and five academic partners. Radhia Cousot is the principal investigator for this project.

8.2.1.2. MemCad

Title: Memory Compositional Abstract Domains

Type: IDEAS

Instrument: ERC Starting Grant (Starting)

Duration: October 2011 - September 2016

Coordinator: INRIA (France)

Others partners: none

See also: http://www.di.ens.fr/ rival/memcad.html

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 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. NSFC Project

8.3.1.1. NSFC

Title: Analysis and Verification of Dependable Cyber-Physical Software

Type: National Natural Science Foundation of China (NSFC)

Duration: January 2012 - December 2016

Coordinator: National University of Defense Technology (China)

Others partners: National University of Defense Technology (China), Seoul National University (Korea)

Abstract: The project addresses analysis and verification issues related to dependability properties of Cyber Physical Systems (CPS) software: safety (such as the numerical or and memory related runtime errors), quantitative properties (such as the worst-case execution time, upper bound of the memory consumption, etc.), stability and robustness (due to intrinsic uncertainty of CPS), as well as properties of hybrid system (which provides a model for describing the coordination of computation and physical, discrete and continuous processes). The project is expected to advance the analysis and verification methodology for dependable CPS software so as to contribute to the dependability assurance of CPS software in mission critical applications. Patrick Cousot is the principal investigator for this project.

8.3.1.2. Visiting professors

Yanjun Wen is associate professor at the Department of Computer Science and Technology, College of Computer, National University of Defense Technology, Changsha, P. R. China. He is visiting the team from June 2011 to May 2012 and is interested in the static analysis of parallel software by abstract interpretation.

Roberto Giacobazzi, professor at the University of Verona, Italy, visited in spring 2011.

Andreas Podelski, professor at the University of Freiburg, Germany, visited in fall 2011.

8.3.1.3. Internship

Marie Pelleau is a third year PhD student from the University of Nantes (France) under the supervision of Frédéric Benhamou, Pascal Van Hentenryck, and Charlotte Truchet. She spent one month (November 2011) in the team, under the supervision of Antoine Miné, on the application of numerical abstract domains (and in particular, the Apron library, 5.1) to constraint programming.

David Delmas is an engineer at Airbus France on educational leave to pursue the 2nd year of the Parisian Master of Research in Computer Science (MPRI) and a visitor in the team from September 2011 to August 2012.

Suzanne Renard is a third year student at École des Mines de Paris (France). She spent six months (September 2010 to February 2011) in the team, under the supervision of Xavier Rival; she was working on the extension of the XISA shape analysis frameworks in order to express set properties.

9. Dissemination

9.1. Animation of the scientific community

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-Software (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

Patrick Cousot is director of studies in computer science at ENS and member of the *commission de spécialistes* (hiring committee) of ENS.

Patrick Cousot, Antoine Miné 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 de Cachan, antenne de Bretagne (Ker Lann, France).

Xavier Rival was a member of the *comité de sélection* (hiring committee) to hire an assistant professor at the Université de Paris 7.

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.

Patrick Cousot was member of the program committees of the 32th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2011 ERC) , San Jose, CA, USA, June 4-8, 2011; the 12th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2011), Austin, TX, USA, January 23-25, 2011; the 14th ACM International Conference on Hybrid Systems (HSCC 2011), Chicago, IL, USA, April 11-14, 2011; Verified Software: Theories, Tools and Experiments (VSTTE 2012), Philadelphia, USA, January 28-29, 2012; the 19th International Static Analysis Symposium (SAS'12), Deauville, France; the 15th ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2012), Beijing, China, April 17-19, 2012.

— 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 is the program committee chair of the 40th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 2013), Rome, Italy, January 23-25, 2013.

Radhia Cousot was member of the program committees of the 21st European Symposium on Programming (ESOP 2011), Saarbrücken, Germany, March 26-April 3, 2011; the 18th International Static Analysis Symposium (SAS'11), Venice, Italy, September 14-16, 2011; the 38th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 2011), Austin, Texas, USA, January 26-28, 2011.

— Jérôme Feret is a member of the editorial board of the Frontiers in Genetics journal.

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 2nd SASB (2011) and is co-program committee chair of the 3rd SASB (2012).

Jérôme Feret was member of the program committee of the 2nd International Workshop on Interactions between Computer Science and Biology (CS2Bio 2011), the 9th International Conference on Computational Methods in Systems Biology (CMSB 2011), the 9th Asian Symposium on Programming Languages (APLAS 2011), the 4th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2012). He will be a member of the International Symposium on Foundations of Health Information Engineering and System (FHIES 2012).

— Antoine Miné was member of the program committee of the 18th International Static Analysis Symposium (SAS'11), the third Workshop on Numerical and Symbolic Abstract Domains (NSAD'11), and the First International Workshop on Safety and Security in Cyber-Physical Systems (SSCPS'11).

Antoine Miné will be program committee co-chair and general chair of the 19th International Static Analysis Symposium (SAS'12), Deauville, France, general chair of the 4th International Workshop on Numerical and Symbolic Abstract Domains (NSAD'12), the 3rd International Workshop on Static Analysis and Systems Biology (SASB'12), and the 3rd International Workshop on Tools for Automatic Program AnalysiS (TAPAS'12), Deauville, France, and member of the program committee of the Second International Workshop on Safety and Security in Cyber-Physical Systems (SSCPS'12), Gaithersburg, Maryland, USA.

— Xavier Rival was member of the program committee the Conferences on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2011), Saarbrücken, Germany, March 26-April 3, 2011.

Xavier Rival is a member of the program committee the European Symposium On Programming (ESOP 2012).

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

9.1.4. Jury of PhD and Habilitation

— Patrick Cousot was in the jury of the habilitation of Xavier Rival, (ENS, Paris, France, June 24, 2011).

— Jérôme Feret was in the jury of the PhD thesis of Loïc Paulevé (IRCCyn, Nantes, France, October 6, 2011).

— Antoine Miné was in the jury of the PhD thesis of Khalil Ghorbal (CEA, France, July 28, 2011).

9.1.5. Participation in Conferences

CMSB: Ninth International Conference on Computational Methods in Systems Biology (Paris, France, 21–23 September 2011).
Ferdinanda Camporesi, Jérôme Feret, and Alessandro Romanel attended the workshop. Jérôme Feret chaired a session.

ESOP: European Symposium on Programming (Saarbrücken, Germany, 30 March – 1st April 2011)
Patrick Cousot, Radhia Cousot, Antoine Miné and Xavier Rival attended the conference. Antoine Miné gave a talk on the static analysis of parallel programs [22].

FOSSACS: 14th International Conference on Foundations of Software Science and Computation Structures (Saarbrücken, Germany, 29–31 March 2011)
Patrick Cousot, Radhia Cousot attended the conference. Patrick Cousot gave a talk on the reduced product of abstract domains and the combination of decision procedures [20].

MecBIC: International Workshop on Membrane Computing and Biologically Inspired Process Calculi (Fontainebleau, France, 23 August 2011).
Jérôme Feret and Alessandro Romanel attended the workshop.

MFPS: International Conference on Mathematical Foundations of Programming Semantics (Pittsburg, Pennsylvania, USA, 25–28 May 2011).
Jérôme Feret attended the conference and gave an invited talk on model reduction of differential models [13].

NSAD: Second International Workshop on Numerical and Symbolic Abstract Domains (Venice, Italy, 13 September 2011).
Antoine Miné, Patrick Cousot, Radhia Cousot, and Xavier Rival attended the workshop.

POPL: ACM Symposium on Principles of Programming Languages (Austin, Texas, USA, 26–28 January 2011).
Patrick Cousot, Radhia Cousot and Xavier Rival attended the conference [18]. Xavier Rival gave a talk on Calling context abstraction with shapes [23].

RAIM: 4ème Rencontres Arithmétique de l'Informatique Mathématique (Perpignan, France, 7–10 February 2011)
Antoine Miné attended and gave a talk on the static analysis of numerical programs manipulating floating-point numbers.

SAS: 18th International Static Analysis Symposium (Venice, Italy, 14–16 September 2011).
Ferdinanda Camporesi, Patrick Cousot, Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival, and Matteo Zanioli attended the conference. Patrick Cousot gave an invited talk on Combining Algebraic Domains and Logical Theories by the Reduced Product. Jérôme Feret gave an invited talk on model reduction of differential models [14]. Antoine Miné chaired a session.

SASB: International Workshop on Static Analyis and Systems Biology (Venice, Italy, 13 September 2011).
Ferdinanda Camporesi and Jérôme Feret attended to the workshop. Jérôme Feret co-chaired the workshop and chaired all the sessions.

TACAS: 17th International Conference on Tools for Automatic Construction and Analysis of Systems (Saarbrücken, Germany, 29–31 March 2011)
Xavier Rival attended the conference and chaired a session.

TAPAS: Second International Workshop on Tools for Automatic Program Analysis (Venice, Italy, France, 17 September 2011).
Antoine Miné and Xavier Rival attended the workshop.

VMCAI: International Conference on Verification, Model Checking and Abstract Interpretation (Austin, Texas, USA, 23–25 January 2011).
Patrick Cousot, Radhia Cousot attended the conference. Patrick Cousot gave a talk on precondition inference from intermittent assertions and application to contracts on collections [19].

9.1.6. Invitations and Participation in Seminars

— Ferdinanda Camporesi gave a talk on model reduction of signaling pathways at the Semantics and Abstraction Interpretation Seminar (ENS, Paris, France).

— Patrick Cousot gave a talk on Unifying proof theoretic/logical and algebraic abstractions for inference and verification, NSF CMACS Meeting, University of Maryland, College Park, MD, USA, April 28-29, 2011; on Theories, Solvers and Static Analysis by Abstract Interpretation, ASCERT Meeting, ENS Paris, France, November 30, 2011; on Program verification by abstract interpretation, NSF CMACS Industry Workshop on Verification of Embedded Control Systems, October 20, 2011, Carnegie Mellon University, Pittsburgh, PA.

— Patrick Cousot and Radhia Cousot gave a talk on Method Refactoring by Abstract Interpretation, MSR Talk Series, Microsoft Research, Redmond, WA, USA, September 2, 2011; on Theories, Solvers and Static Analysis by Abstract Interpretation, MSR Talk Series, Microsoft Research, Redmond, WA, USA. August 12th, 2011.

— Jérôme Feret gave a talk on the ASTRÉE analyzer at the Programming Methodology group at ETH Zürich (Zürich, Switzerland) and some talks on model reduction for signaling pathways at the Bison group at ETH Zürich (Zürich, Switzerland), at the Focus group at the University of Bologna (Bologna, Italy), at the ANR-SYMBIOTIC meeting (IBISC, Evry, France), at the SysBio meeting of the Systems Biology of cancer group at the Institut Curie (Paris, France).

— Antoine Miné gave a talk on the static analysis of parallel programs at the 68NQRT Seminar, IRISA and INRIA Rennes (France) on the 26 May 2011, at the Semantics and Abstraction Interpretation Seminar, École normale supérieure (Paris, France) on the 18 November 2011, and at IMDEA-Software (Madrid, Spain) on the 12 December 2011.

— Xavier Rival was invited to give a talk on Perspective for compiler certification in avionics at the "Compiler Optimization meets Compiler Verification" Workshop (COCV) at ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Xavier Rival gave a talk on Abstract domains for the static analysis of programs manipulating complex data-structures at Seoul National University (Seoul, Korea), on the 26th August, 2011.

9.2. Teaching

Licence :

- Mathematics, 20h, L1, Licence Frontiers in Life Sciences, Université Paris-Descartes, France.
- Introduction to static analysis, 8h, L3, École des Mines de Paris, France.
- Introduction to algorithmics, 40h, L2, École Polytechnique, Palaiseau, France.
- Algorithmics and programming, 40h, L3, École Polytechnique, Palaiseau, France.

Master :

- Computational Biology, 6h, M1, Interdisciplinary Approaches to Life Science (AIV) Master Program, Université Paris-Descartes, France
- Abstract interpretation: application to verification and static analysis, 48h, niveau M2, Parisian Master of Research in Computer Science (MPRI), École normale supérieure, France.
- Rule-based modeling and application to biomolecular networks, 8h, M1-M2, Master of Fundamental Research in Computer Science (MIF), École normale supérieure de Lyon, France

Doctorat :

- Abstract Interpretation and its Applications, 19h, University of Bologna / University of Padova, Italy.
- Abstract Interpretation-based Tool Construction for Software Verification, 8th LASER Summer School on Software Engineering (LASER 2011), Elba Island, Italy, September 4-10, 2011.

PhD & HdR :

HdR :

- Xavier Rival, Abstract domains for the analysis of programs manipulating complex data-structures [8], École Normale Supérieure, June, 24th, 2011.

PhD in progress :

- Mehdi Bouaziz, November 2011, Patrick Cousot, 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).
- Tie Cheng, Static analysis of spreadsheet macros, October 2011, Xavier Rival, École Polytechnique
- Vincent Laviron, October 2009, Patrick Cousot, École Normale Supérieure.
- Antoine Toubhans, Combination of shape abstract domains, October 2011, Xavier Rival, École Doctorale de Paris Centre
- Caterina Urban, November 2011, Radhia Cousot, École Normale Supérieure.
- Matteo Zanioli, October 2008, Radhia Cousot (co-directed thesis with Agostino Cortesi, University of Venezia).

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 Institue of Aeronautics and Astronautics (AIAA Infotech@Aerospace 2010)", Atlanta, Georgia, USA, American Institue 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, p. 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, p. 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, p. 5–20.

[6] A. MINÉ. *The Octagon Abstract Domain*, in "Higher-Order and Symbolic Computation", 2006, vol. 19, p. 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, p. 1–13.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] X. RIVAL. *Abstract domains for the analysis of programs manipulating complex data-structures*, École Normale Supérieure, June 2011, Habilitation à Diriger des Recherches.

Articles in International Peer-Reviewed Journal

[9] J. BERTRANE, P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, X. RIVAL. *Static analysis by abstract interpretation of embedded critical software*, in "ACM SIGSOFT Software Engineering Notes", 2011, vol. 36, n^{o} 1, p. 1-8.

[10] A. CORTESI, M. ZANIOLI. *Widening and narrowing operators for abstract interpretation*, in "Computer Languages, Systems and Structures", 2011, vol. 37, n^{o} 1, p. 24 - 42, http://dx.doi.org/10.1016/j.cl.2010.09.001.

[11] P. COUSOT, R. COUSOT. *Grammar semantics, analysis and parsing by abstract interpretation*, in "Theoretical Computer Science", 2011, vol. 412, n^{o} 44, p. 6135-6192.

[12] J. FERET, T. HENZINGER, H. KOEPPL, T. PETROV. *Lumpability Abstractions of Rule-based Systems*, in "Theorerical Computer Science", 2012, to appear, http://dx.doi.org/10.1016/j.tcs.2011.12.059.

Invited Conferences

[13] F. CAMPORESI, J. FERET. *Formal reduction for rule-based models*, in "Post-proceedings of the the 27th Conference on the Mathematical Foundations of Programming Semantics - (MFPS'11)", Pittsburgh, United States, M. MISLOVE, J. OUAKNINE (editors), Electronic Notes in Theoretical Computer Science, Elsevier, September 2011, vol. 276, p. 29-59 [*DOI :* 10.1016/J.ENTCS.2011.09.014], http://hal.inria.fr/inria-00636850/en.

[14] J. FERET. *Formal Model Reduction*, in "Proceedings of the 18th International Static Analysis Symposium (SAS'11)", Venice, Italy, E. YAHAV (editor), Lecture Notes in Computer Science, 2011, vol. 6887, p. 6–6 [*DOI :* 10.1007/978-3-642-23702-7_5], http://hal.inria.fr/inria-00626640/en.

International Conferences with Proceedings

[15] L. CHEN, A. MINÉ, J. WANG, P. COUSOT. *Linear Absolute Value Relation Analysis*, in "Proceedings of the 20th European Symposium on Programming (ESOP'11)", G. BARTHE (editor), Lecture Notes in Computer Science, Springer, 2011, vol. 6602, p. 156–175, http://hal.inria.fr/hal-00648039/.

[16] A. CORTESI, M. ZANIOLI. *Information Leakage Analysis by Abstract Interpretation*, in "Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science", Novy Smokovec Slovakia, Lecture Notes in Computer Science, Springer, 2011, vol. 6543, p. 545–557.

[17] P. COUSOT, R. COUSOT. *An Abstract Interpretation Framework for Termination*, in "Proceedings of the 39th Annual ACM Symposium on Principles Of Programming Languages (POPL'12)", Philadelphia, PA, ACM Press, January 25–27 2012.

[18] P. COUSOT, R. COUSOT, F. LOGOZZO. *A Parametric Segmentation Functor for Fully Automatic and Scalable Array Content Analysis*, in "Proceedings of the 38th Annual ACM Symposium on Principles Of Programming Languages (POPL'11)", Austin, Texas, United States, ACM Press, 2011, http://hal.inria.fr/inria-00543874/en.

[19] P. COUSOT, R. COUSOT, F. LOGOZZO. *Precondition Inference from Intermittent Assertions and Application to Contracts on Collections*, in "Proceedings of the 12th Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'11)", Austin, Texas, United States, R. JHALA, D. SCHMIDT (editors), Springer-Verlag, 2011, http://hal.inria.fr/inria-00543881/en.

[20] P. COUSOT, R. COUSOT, L. MAUBORGNE. *The Reduced Product of Abstract Domains and the Combination of Decision Procedures*, in "Proceedings of Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011", Saarbrücken, Germany, M. HOFMANN (editor), Lecture Notes in Computer Science, Springer, March-April 2011, vol. 6604, p. 456-472.

[21] P. COUSOT, M. MONERAU. *Probabilistic Abstract Interpretation*, in "Proceedings of the 21th European Symposium on Programming (ESOP'12)", H. SEIDL (editor), Lecture Notes in Computer Science, Springer, March 2012, to appear.

[22] A. MINÉ. *Static Analysis of Run-Time Errors in Embedded Critical Parallel C Programs*, in "Proceedings of the 20th European Symposium on Programming (ESOP'11)", G. BARTHE (editor), Lecture Notes in Computer Science, Springer, 2011, vol. 6602, p. 398–418, http://hal.inria.fr/hal-00648038.

[23] X. RIVAL, B.-Y. E. CHANG. *Calling Contexts Abstraction with Shapes*, in "Proceedings of the 38th Annual ACM Symposium on Principles Of Programming Languages (POPL'10)", Austin, Texas, ACM Press, January 26–28, 2011.

[24] M. ZANIOLI, P. FERRARA, A. CORTESI. *SAILS: static analysis of information leakage with Sample*, in "Proceedings of the 27th ACM Symposium on Applied Computing (SAC'12)", Riva del Garda, Italy, ACM Press, 2012, to appear.

Scientific Books (or Scientific Book chapters)

[25] J. BERTRANE, J. FERET, P. COUSOT, R. COUSOT, A. MINÉ, X. RIVAL, L. MAUBORGNE. *L'analyseur statique Astrée*, in "Utilisations industrielles des techniques formelles : interprétation abstraite", J.-L. BOULANGER (editor), Hermes-Lavoisier, June 2011, p. 67–113, http://hal.inria.fr/inria-00636877/en.

Books or Proceedings Editing

[26] J. FERET, A. LEVCHENKO (editors). *Static Analysis and Systems Biology – 1st International Workshop, SASB 2010, Perpignan, France, September 13, 2010. PostProceedings*, Electronic Notes in Theoretical Computer Science, Elsevier, 2011, vol. 272.

References in notes

[27] 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, p. 7–9.

[28] 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, Grenoble, France, 21 March 1978.

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

[30] 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, p. 303–342.

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

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

[33] 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, p. 421–505.

[34] 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, p. 359–366.

[35] 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.

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

[37] 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, p. 238–252.

[38] 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, p. 269–282.

[39] 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, p. 119–133.

[40] 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, p. 243–271.

[41] P. COUSOT, R. COUSOT. *Higher-Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection and PER Analysis of Functional Languages), invited paper*, in "Proceedings of the 1994 International Conference on Computer Languages", Toulouse, France, IEEE Computer Society Press, Los Alamitos, California, 16–19 May 1994, p. 95–112.

[42] 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, p. 21–30.

[43] 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.

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

[45] 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, p. 362–381, http://hal.inria.fr/hal-00520112, http://hal.inria.fr/hal-00520112.

[46] 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, p. 17–41, http://hal.archives-ouvertes.fr/hal-00164297/en/.

[47] 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, p. 139-157 [*DOI :* 10.1.1.139.5120], http://hal.inria.fr/inria-00528409/en/.

[48] 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, p. 83-97 [*DOI :* 10.1007/978-3-540-78163-9_11], http://hal.inria.fr/inria-00528352/en/.

[49] V. DANOS, C. LANEVE. *Formal Molecular Biology*, in "Theoretical Computer Science", 10 2004, vol. 325, n^{o} 1, p. 69-110 [*DOI :* 10.1016/J.TCS.2004.03.065], http://hal.archives-ouvertes.fr/hal-00164591/en/.

[50] 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, 037108, http://hal.inria.fr/hal-00520128, http://hal.inria.fr/hal-00520128.