IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2012

Project-Team ABSTRACTION

Abstract Interpretation and Static Analysis

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

RESEARCH CENTER
Paris - Rocquencourt

THEME
Programs, Verification and Proofs

Table of contents

1. Members **1**
2. Overall Objectives **1**
2.1. Overall Objectives 1
2.2. Highlights of the Year 2
3. Scientific Foundations **2**
3.1. Abstract Interpretation Theory 2
3.2. Formal Verification by Abstract Interpretation 2
3.3. Advanced Introductions to Abstract Interpretation 3
4. Application Domains **3**
4.1. Certification of Safety Critical Software 3
4.2. Abstraction of Biological Cell Signaling Networks 4
5. Software **4**
5.1. The Apron Numerical Abstract Domain Library 4
5.2. The Astrée Static Analyzer of Synchronous Software 5
5.3. The AstréeA Static Analyzer of Asynchronous Software 6
5.4. The OpenKappa modeling plateform 7
5.5. Translation Validation 7
5.6. Zarith 8
6. New Results **8**
6.1. Analysis of Biological Pathways 8
6.1.1. Semantics 8
6.1.2. Semantics and causality 8
6.1.3. Case study: Combinatorial drift in yeast model 9
6.1.4. Automatic Reduction of Stochastic Semantics 9
6.2. Leakage Analysis 9
6.3. Termination 10
6.4. Probabilistic Abstract Interpretation 10
6.5. Formal Verification by Abstract Interpretation 10
6.6. Static Analysis of Parallel Software 11
6.7. Static Analysis of Bit-Level Machine Integer and Floating-Point Operations 11
6.8. Inferring Sufficient Conditions with Backward Polyhedral Under-Approximations 12
6.9. A Constraint Solver Based on Abstract Domains 12
6.10. Automatic Inference of Necessary Preconditions 12
6.11. Inference of Necessary Field Conditions with Abstract Interpretation 13
6.12. TreeKs: A Functor to Make Numerical Abstract Domains Scalable 13
6.13. An Abstract Domain to Infer Types over Zones in Spreadsheets 13
6.14. Hierarchical Abstraction of Dynamic Structures 13
6.15. Reduced Product Combination of Abstract Domains for Shapes 14
7. Bilateral Contracts and Grants with Industry **14**
7.1. Contracts with Industry 14
7.2. Grants with Industry 14
7.2.1.1. Ascert 14
7.2.1.2. Sardanes 15
8. Partnerships and Cooperations **15**
8.1. National Initiatives 15
8.1.1.1. AbstractCell 15
8.1.1.2. AstréeA 16
8.1.1.3. Verasco 16
8.2. European Initiatives 16

8.2.1.1. MBAT 16
8.2.1.2. MemCad 17
8.3. International Research Visitors 17
9. Dissemination **18**
9.1. Scientific Animation 18
9.1.1. Academy Members, Professional Societies 18
9.1.2. Collective Responsibilities 18
9.1.3. Editorial Boards and Program Committees 18
9.1.4. Participation in Conferences 19
9.1.5. Invitations and Participation in Seminars 21
9.2. Teaching - Supervision - Juries 21
9.2.1. Teaching 21
9.2.2. Supervision 22
9.2.3. Juries 22
9.3. Popularization 22
10. Bibliography **22**

Project-Team ABSTRACTION

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

The project-team ABSTRACTION *is located at the École normale supérieure, Paris.*

Creation of the Project-Team: January 01, 2008 .

1. Members

Research Scientists

Radhia Cousot [Senior Researcher, CNRS, HdR]
Jérôme Feret [Junior Researcher, Inria Paris–Rocquencourt]
Antoine Miné [Junior Researcher, CNRS]
Xavier Rival [Team leader by interim, Junior Researcher, Inria Paris–Rocquencourt, HdR]

Faculty Member

Patrick Cousot [Team leader, Professor, ENS, HdR]

PhD Students

Mehdi Bouaziz [Nov. 2012 —]
Ferdinanda Camporesi
Vincent Laviron
Cheng Tie
Antoine Toubhans
Caterina Urban
Matteo Zanioli [— March 2012]

Post-Doctoral Fellows

Norman Ferns [Mars. 2012 —]
Luca Grieco [Oct. 2012 — Dec. 2012]
Jonathan Hayman [— Oct. 2012]
Pascal Sotin [— Aug. 2012]
Arnaud Spiwack [Sept. 2012 —]

Visiting Scientists

David Delmas [— Aug. 2012]
Yanjun Wen [— May 2012]

Administrative Assistants

Joëlle Isnard [Administrative Head DI, ENS]
Marine Meyer [Inria]

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

Antoine Miné was the program cochair and the local organizer of the 19th international statis analysis symposium (SAS 2012) in Deauville, September 11–13 2012 and Radhia Cousot is the program chair the 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2013) in Roma, January 23–25 2013.

3. Scientific Foundations

3.1. Abstract Interpretation Theory

The abstract interpretation theory [41], [32], [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], [34], or parallel [36] with shared memory [33], [35], [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. [39], [40], [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* [40], [47] and automatic tools for *program analysis* (computing an abstract semantics) and *program verification* (proving that an abstract semantics satisfies an abstract specification) [37].

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 [37], 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" [38] 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

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 [31], [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

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

5.1. The Apron Numerical Abstract Domain Library

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

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).

In 2012, APRON has been used in several researches conducted within or in collaboration with the Abstraction project-team: the design of a sufficient-condition generator [23] and the design of a constraint solver based on abstract domains [25].

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.

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

ASTRÉE is a static analyzer for sequential programs based on abstract interpretation [41], [32], [42], [34].

The ASTRÉE static analyzer [31], [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 scientifique leader, correspondent], Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival.

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], [36]. 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 1,200 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 (Section 8.1.1.2).

5.4. The OpenKappa modeling plateform

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

Causal traces, Model reduction, Rule-based modelling, 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 [52], a static analyzer [51] (for debugging models), a simulator [50], a compression tool for causal traces [49],[20], and a model reduction tool [4], [48], [53].

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 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).

5.5. Translation Validation

Participant: Xavier Rival [correspondent].

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 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].

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. 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).

6.1.1. Semantics

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

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 [21], 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. Semantics and causality

Participants: Vincent Danos [University of Edinburgh], Jérôme Feret, Walter Fontana [Harvard Medical School], Russ Harmer [Paris VII], Jonathan Hayman, Jean Krivine [Paris VII], Chris Thompson-Walsh [University of Cambridge], Glynn Winskel [University of Cambridge].

In [20], we introduce a novel way of constructing concise causal histories (pathways) to represent how specified structures are formed during simulation of systems represented by rulebased models. This is founded on a new, clean, graph-based semantics introduced in the first part of this paper for Kappa, a rule-based modelling language that has emerged as a natural description of protein-protein interactions in molecular biology. The semantics is capable of capturing the whole of Kappa, including subtle side-effects on deletion of structure, and its structured presentation provides the basis for the translation of techniques to other models. In particular, we give a notion of trajectory compression, which restricts a trace culminating in the production of a given structure to the actions necessary for the structure to occur. This is central to the reconstruction of biochemical pathways due to the failure of traditional techniques to provide adequately concise causal histories, and we expect it to be applicable in a range of other modelling situations.

6.1.3. Case study: Combinatorial drift in yeast model

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

The assembly of molecular machines and transient signaling complexes does not typically occur under circumstances in which the appropriate proteins are isolated from all others present in the cell. Rather, assembly must proceed in the context of large-scale protein-protein interaction (PPI) networks that are characterized both by conflict and combinatorial complexity. Conflict refers to the fact that protein interfaces can often bind many different partners in a mutually exclusive way, while combinatorial complexity refers to the explosion in the number of distinct complexes that can be formed by a network of binding possibilities.

In [9], we use computational models so as to explore the consequences of these characteristics for the global dynamics of a PPI network based on highly curated yeast two-hybrid data. The limited molecular context represented in this data-type translates formally into an assumption of independent binding sites for each protein. The challenge of avoiding the explicit enumeration of the astronomically many possibilities for complex formation is met by a rule-based approach to kinetic modeling. Despite imposing global biophysical constraints, we find that initially identical simulations rapidly diverge in the space of molecular possibilities, eventually sampling disjoint sets of large complexes. We refer to this phenomenon as “compositional drift". Since interaction data in PPI networks lack detailed information about geometric and biological constraints, our study does not represent a quantitative description of cellular dynamics. Rather, our work brings to light a fundamental problem (the control of compositional drift) that must be solved by mechanisms of assembly in the context of large networks. In cases where drift is not (or cannot be) completely controlled by the cell, this phenomenon could constitute a novel source of phenotypic heterogeneity in cell populations.

6.1.4. Automatic Reduction of Stochastic Semantics

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

Biology, Protein-protein interaction networks, Stochastic semantics, Verification.

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 each state with the same abstraction is equiprobable at initial state, each state with the same abstraction is equiprobable at any time t.

In [10], we have formalized the model reduction framework for the stochastic semantics and we have established the relationships with the notions of lumpability, and bisimulation.

In [13], we have showed that the reduced models can be expressed in Kappa, and we have provided a procedure to do it.

6.2. Leakage Analysis

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

Abstract interpretation, Information leakage analysis, Object-oriented software, Static analysis.

In [28], 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.

6.3. Termination

Participants: Patrick Cousot, Radhia Cousot.

Abstract interpretation, Computational induction, Induction, Proof, Static analysis, Semantic structural induction, Syntactic structural induction, Termination, Variant function, Verification.

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.

6.4. Probabilistic Abstract Interpretation

Participants: Patrick Cousot, Michaël Monerau.

Abstract interpretation, Probabilistic systems, Static analysis.

Abstract interpretation has been widely used for verifying properties of computer systems. In [19], we present a way to extend this framework to the case of probabilistic systems.

The probabilistic abstraction framework that we propose allows us 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 define the concrete probabilistic semantics and propose different ways to abstract them. We provide examples illustrating the expressiveness and effectiveness of our approach.

6.5. Formal Verification by Abstract Interpretation

Participant: Patrick Cousot.

Abstract interpretation, Abstraction, Aerospace, Certification, Cyber-physical system, Formal Method, Mission-critical system, Runtime error, Safety-critical system, Scalability, Soundness, Static Analysis, Validation, Verification.

Abstract interpretation is a theory of abstraction and constructive approximation of the mathematical structures used in the formal description of programming languages and the inference or verification of undecidable program properties. Developed in the late seventies with Radhia Cousot, it has since then been considerably applied to many aspects of programming, from syntax, to semantics, and proof methods where abstractions are sound and complete but incomputable to fully automatic, sound but incomplete approximate abstractions to solve undecidable problems such as static analysis of infinite state software systems, contract inference, type inference, termination inference, model-checking, abstraction refinement, program transformation (including watermarking), combination of decision procedures, security, malware detection, etc.

This last decade, abstract interpretation has been very successful in program verification for mission- and safety-critical systems [12]. An example is ASTRÉE which is a static analyzer to verify the absence of runtime errors in structured, very large C programs with complex memory usages, and involving complex boolean as well as floating-point computations (which are handled precisely and safely by taking all possible rounding errors into account), but without recursion or dynamic memory allocation. Astrée targets embedded applications as found in earth transportation, nuclear energy, medical instrumentation, aeronautics and space flight, in particular synchronous control/command such as electric flight control or more recently asynchronous systems as found in the automotive industry. Astrée is industrialized by AbsInt Angewandte Informatik GmbH.

6.6. Static Analysis of Parallel Software

Participant: Antoine Miné.

Abstract interpretation, Embedded software, Parallel software, Rely/guarantee analysis, Run-time errors, Static analysis.

We present in [11] the theoretical foundation and the latest experimental evaluation of ASTRÉEA (5.3), a static analyzer prototype based on abstract interpretation to check for run-time errors in multi-threaded embedded critical C programs. Our method is based on a slightly modified non-parallel analysis that, when analyzing a thread, applies and enriches an abstract set of thread interferences. An iterator then re-analyzes each thread in turn until interferences stabilize. We prove the soundness of our method with respect to the sequential consistency semantics, but also with respect to a reasonable weakly consistent memory semantics. We also show how to take into account mutual exclusion and thread priorities through a partitioning over an abstraction of the scheduler state. This work is an extension of [54], complete with a full formalization and soundness proofs.

In [24], we express rely/guarantee methods in constructive form as an abstract interpretation of the interleaving trace semantics. We also restate the analysis presented in [11] as a further abstraction of rely/guarantee. This theoretical work brings a new understanding of the various causes of incompleteness and imprecision in our previous analysis, including the non-relational, input-insensitive, flow-insensitive, and history-insensitive treatment of interferences, and it opens the way to designing more precise analyses.

6.7. Static Analysis of Bit-Level Machine Integer and Floating-Point Operations

Participant: Antoine Miné.

Abstract interpretation, Embedded software, Numerical abstract domains, Run-time errors, Static analysis.

We present in [22] a few lightweight numeric abstract domains to analyze C programs that exploit the binary representation of numbers in computers, for instance to perform "compute-through-overflow" on machine integers, or to directly manipulate the exponent and mantissa of floating-point numbers. On integers, we propose an extension of intervals with a modular component, as well as a bitfield domain. On floating-point numbers, we propose a predicate domain to match, infer, and propagate selected expression patterns. These domains are simple, efficient, and extensible. We have included them into the ASTRÉE (5.2) and ASTRÉEA (5.3) static analyzers to supplement existing domains. Experimental results show that they can improve the analysis precision at a reasonable cost.

6.8. Inferring Sufficient Conditions with Backward Polyhedral Under-Approximations

Participant: Antoine Miné.

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

In [23], we discuss the automatic inference of sufficient pre-conditions by abstract interpretation and sketch the construction of an under-approximating backward analysis. We focus on numeric domains and propose transfer functions, including a lower widening, for polyhedra, without resorting to disjunctive completion nor complementation, while soundly handling non-determinism. A limited proof-of-concept prototype was designed to validate our approach. Planned 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.

6.9. 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].

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

In [25], 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.

6.10. Automatic Inference of Necessary Preconditions

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

Abstract interpretation, Backward analysis, Static analysis, Necessary condition inference,

In [18], 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%.

6.11. Inference of Necessary Field Conditions with Abstract Interpretation

Participants: Mehdi Bouaziz, Manuel Fahndrich [Microsoft Research, Redmond, USA], Francesco Logozzo [Microsoft Research, Redmond, USA].

In [15], we present a new static analysis to infer necessary field conditions for object-oriented programs. A necessary field condition is a property that should hold on the fields of a given object, for otherwise there exists a calling context leading to a failure due to bad object state. Our analysis also infers the provenance of the necessary condition, so that if a necessary field condition is violated then an explanation containing the sequence of method calls leading to a failing assertion can be produced.

When the analysis is restricted to readonly fields, i.e., fields that can only be set in the initialization phase of an object, it infers object invariants. We provide empirical evidence on the usefulness of necessary field conditions by integrating the analysis into cccheck, our static analyzer for .NET.

Robust inference of readonly object field invariants was the #1 request from cccheck users.

6.12. TreeKs: A Functor to Make Numerical Abstract Domains Scalable

Participant: Mehdi Bouaziz.

Relational numerical abstract domains do not scale up. To ensure a linear cost of abstract domains, abstract interpretation-based tools analyzing large programs generally split the set of variables into independent smaller sets, sometimes sharing some non-relational information. In [14], we present a way to gain precision by keeping fully expressive relations between the subsets of variables, whilst retaining a linear complexity ensuring scalability.

6.13. An Abstract Domain to Infer Types over Zones in Spreadsheets

Participants: Cheng Tie, Xavier Rival.

abstract domains, spreadsheet script languages In [16], we proposed an abstract domain for the abstraction of spreadsheet contents.

Spreadsheet languages are very commonly used, by large user bases, yet they are error prone. However, many semantic issues and errors could be avoided by enforcing a stricter type discipline. As declaring and specifying type information would represent a prohibitive amount of work for users, we propose an abstract interpretation based static analysis for spreadsheet programs that infers type constraints over zones of spreadsheets, viewed as two-dimensional arrays. Our abstract domain consists in a cardinal power from a numerical abstraction describing zones in a spreadsheet to an abstraction of cell values, including type properties. We formalize this abstract domain and its operators (transfer functions, join, widening and reduction) as well as a static analysis for a simplified spreadsheet language. Last, we propose a representation for abstract values and present an implementation of our analysis.

6.14. Hierarchical Abstraction of Dynamic Structures

Participants: Pascal Sotin, Xavier Rival.

abstract domains, shape analysis, domain combination In [26], we designed a hierarchical shape abstract domain for the abstraction of complex data structures found in embedded softwares.

We propose a hierarchical shape abstract domain, so as to infer structural invariants of dynamic structures such as lists living inside static structures, such as arrays. This programming pattern is often used in safety critical embedded software that need to “allocate” dynamic structures inside static regions due to dynamic memory allocation being forbidden in this context. Our abstract domain precisely describes such hierarchies of structures. It combines several instances of simple shape abstract domains, dedicated to the representation of elementary shape properties, and also embeds a numerical abstract domain. This modular construction greatly simplifies the design and the implementation of the abstract domain. We provide an implementation, and show the effectiveness of our approach on a problem taken from a real code.

6.15. Reduced Product Combination of Abstract Domains for Shapes

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

abstract domains, shape analysis, reduced product In [27], we proposed a notion of reduced product for shape abstractions.

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.

7. Bilateral Contracts and Grants with Industry

7.1. 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.

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 2012

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

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. FP7 Projects

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

8.3. International Research Visitors

8.3.1. Visits of International Scientists

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 has visited 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 the Team in May 2012.

Michael Hicks is associate professor at the Department of Computer Science, University of Maryland, USA. He has visited the team in October 2012 and is interested in abstract interpretation, software security, and differential privacy.

Tatjana Petrov is a PhD student at ETH Zürich. She has visited the team in February 2012 and is interested in the model reduction of stochastic systems.

8.3.1.1. Internships

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). He has visited the team from September 2011 to August 2012.

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-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.

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 Verified Software: Theories, Tools and Experiments (VSTTE 2012), Philadelphia, USA, January 28-29, 2012; Hybrid Systems: Computation and Control (HSCC 2012), Beijing, China, April 17-19, 2012; the Programming Languages day 2012, IBM Thomas J. Watson Center, New York, USA, June 29, 2012; the 19th Static Analysis Symposium (SAS 2012), Deauville, France, September 11-13, 2012; Hybrid Systems: Computation and Control (HSCC 2013), Philadelphia, PA, USA, April 9-11, 2013.

Patrick Cousot was the co-program committee chair of the Workshop on Systems Biology and Formals Methods (SBFM 2012), New York, NY, USA, March 29-30, 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 committee of the 13th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2012), January 22-24, 2012, Philadelphia, USA and Radhia Cousot is a member 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.

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 3rd International Workshop on Static Analysis and Systems Biology (SASB 2012), Deauville, France, September 10, 2012; and of the Workshop on Systems Biology and Formals Methods (SBFM 2012), New York, NY, USA, March 29-30, 2012 and Jérôme Feret is co-program committee chair of the 4th International Workshop on Static Analysis and Systems Biology (SASB 2013), Seattle, WA, USA, June 2013.

Jérôme Feret was member of the program committees of the 4th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2012), St Maarten, Netherlands Antilles, March 25-30, 2012; the 2nd International Workshop on Interactions between Computer Science and Biology (CS2Bio 2012), Stockholm, Sweden, June 16, 2012; the International Symposium on Foundations of Health Information Engineering and System (FHIES 2012), Paris, France, August 27-28, 2012; the 40th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2013 ERC), Rome, Italy, January 23-25, 2013. Jérôme Feret is member of the program committee of 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.

— 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.

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

Antoine Miné was co-chair of the program committee of the 19th International Static Analysis Symposium (SAS 2012), Deauville, France, 11–13 September 2012, and local organizer of SAS 2012 and affiliated events NSAD 2012, SASB'12 2012, TAPAS 2012, Deauville, France, 10–14 September 2012.

Antoine Miné was a member of the program committee of the Second International Workshop on Safety and Security in Cyber-Physical Systems (SSCPS 2012); Washington, DC, USA, 20–22 June 2012. Antoine Miné is a member of the program committee of the 20th International Static Analysis Symposium (SAS 2013), Seattle, WA, USA, 20–22 June 2013.

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

Xavier Rival was member of the program committee of the program committee the European Symposium On Programming (ESOP 2012), Tallinn, Estonia, March 24 - April 1, 2012; 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.

9.1.4. Participation in Conferences

POPL: 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Philadelphia, USA, January 25–27, 2012.
Patick Cousot attended the symposium and gave a talk on an abstract interpretation framework for termination [17].

Dagstuhl workshop: Dagstuhl Seminar 12051 Analysis of Executables: Benefits and Challenges, Dagstuhl, Germany, January 29–February 3rd, 2012.
Xavier Rival attended the workshop and gave a talk on the verification of programs manipulating complex data-structures.

TSD: 2th Tsingshua Software Day (Tsinghua University, Beijing, China, March 15, 2012).
Patrick Cousot gave a talk on automatic large-scale software verification by abstract interpretation.

SBFM: Workshop on Systems Biology and Formals Methods (New York, NY, USA, March 29-30, 2012).
Patrick Cousot, Jérôme Feret, and Norman Ferns attended to the workshop. Patrick Cousot and Jérôme Feret co-chaired the workshop. Patrick Cousot gave a talk called "A Causual Introduction to Abstract Interpretation" and Jérôme Feret gave a talk on model reduction of differential systems. Patrick Cousot and Jérôme Feret chaired some sessions.

ESOP: European Symposium on Programming (Tallinn, Estonia, March 24 – April 1, 2012)
Xavier Rival attended the conference. Xavier Rival chaired a session.

Bellairs Workshop: Workshop on Theory of Markov Processes (Bellairs Research Institute, Holetown, Barbados, April 2-6, 2012).
Jérôme Feret and Norman Ferns attended to the workshop. Jérôme Feret gave a talk on model reduction of stochastic semantics. Norman Ferns gave a talk on bisimulation metrics for continuous Markov decision processes.

NFM: 4th NASA Formal Methods Symposium (Norfolk, Virginia, April 3-5, 2012).
Patrick Cousot gave a talk on formal verification by abstract interpretation [12].

JCOMP: *Cinquièmes rencontres de la communauté française de compilation* (Rennes, France, 18–20 June 2012).
Antoine Miné attended the workshop and gave a talk.

WING: Fourth International Workshop on Invariant Generation (Manchester, UK, 30 June 2012).
Antoine Miné attended the workshop and gave an invited talk.

FMSB: Workshop on Formal Methods in Systems Biology (Berkeley, USA, July 7-8, 2012).
Jérôme Feret gave a talk on the model reduction of differential systems.

Dagstuhl workshop: Information Flow and Its Applications (Schloss Dagstuhl, Dagstuhl, Germany, August 26-31, 2012).
Jérôme Feret and Jonathan Hayman attended to the workshop. Jérôme Feret gave a talk on model reduction of differential semantics. Jonathan Hayman gave a talk on the semantics of Kappa and causal compression. Jérôme Feret chaired a session.

NSAD: Fourth International Workshop on Numerical and Symbolic Abstract Domains (Deauville, France, September 10, 2012).
Mehdi Bouaziz, Tie Cheng, David Delmas, Antoine Miné, Xavier Rival, Antoine Toubhans, and Caterina Urban attended the workshop. Xavier Rival chaired a session. Mehdi Bouaziz and Antoine Miné gave a talk [14], [23].

SASB: Third International Workshop on Static Analysis and Systems Biology (Deauville, France, September 10, 2012).
Ferdinanda Camporesi, Jérôme Feret, Norman Ferns, and Jonathan Hayman attended to the workshop. Jérôme Feret co-chaired the workshop and chaired half of the sessions.

SAS: 19th International Static Analysis Symposium (Deauville, France, September 11-13, 2012).
Mehdi Bouaziz, Ferdinanda Camporesi, Tie Cheng, David Delmas, Jérôme Feret, Norman Ferns, Jonathan Hayman, Vincent Laviron, Antoine Miné, Xavier Rival, Caterina Urban, and Antoine Toubhans attended the conference. Antoine Miné co-chaired the conference and chaired two sessions. Tie Cheng gave a talk [16].

TAPAS: Third International Workshop on Tools for Automatic Program Analysis (Deauville, France, Italy, France, September 14, 2012).
Ferdinanda Camporesi, Tie Cheng, David Delmas, Jérôme Feret, Norman Ferns, Jonathan Hayman, Antoine Miné, Xavier Rival, Antoine Toubhans, and Caterina Urban attended the workshop.

Dagstuhl workshop: Coalgebraic Information Flow and Its Applications, Schloss Dagstuhl, Dagstuhl, Germany, October 7-12, 2012.
Norman Ferns attended the workshop.

SBE: Systems Biology Europe, Madrid, Spain, October 16-17, 2012.
Jérôme Feret gave a talk on the model reduction of differential systems.

Workshop MSR Workshop on "MSR-Inria joint centre", Cambridge (UK), November 5–6, 2012.
Xavier Rival attended the workshop and chaired a session. Xavier Rival gave a demo of ASTRÉE.

Workshop lif Workshop "Let's imagine the future", Rennes (France), November 7–8, 2012.
Xavier Rival attended the workshop and gave a talk on the verification of embedded programs manipulating complex data-structures.

MOVEP: 10th school of Modelling and Verifying Parallel Processes (Marseille, France, 3–7 Dec. 2012).
Antoine Miné attended the school and gave a talk [24].

APLAS: 10th Asian Symposium on Programming Languages and Systems (Kyoto, Japan, 11–13 Dec. 2012).
Mehdi Bouaziz and Xavier Rival attended to the conference. Mehdi Bouaziz and Xavier Rival gave a talk [15], [26].

FSTTCS: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Hyderabad, India, December 15-17, 2012.
Jonathan Hayman attended the conference and gave a talk [20].

9.1.5. Invitations and Participation in Seminars

— Patrick Cousot gave an invited talk at the Computer Science PhD Day, Aula Magna Silvio Trentin, Universitá Ca' Foscari di Venezia, Italy, March 12th, 2012, and a distinguished lecture at the School of Computer Science of Carnegie Mellon University, Pittsburgh, PA, USA, April 12, 2012.

— Jérôme Feret gave some talks on model reduction for signaling pathways at the IMDEA-Softaware (Madrid, Spain) on the 15th of October, 2012, and at the Working Group of the Symbiose team-project (Rennes, France) on the 15th of November, 2012.

— Jonathan Hayman gave a talk at the Working Group on Computational Biology of the École normale supérieure (Paris, France) on the 28th of February, 2012, a talk at the Working Group of CEA List (Saclay, France) on 10th of April, 2012, and a talk at the seminar of the PPS team (Paris, France) on the 14th of June, 2012, about the semantics of Kappa and its causal compression.

— Xavier Rival gave a talk on "MemCAD, A Modular Abstract Domain for Reasoning on Memory States" at Microsoft Research (Redmond, USA) on the 23rd of July, 2012. Xavier Rival gave a talk on "Abstract domains for shape analysis" at CEA, on the 18th of February, 2012. Xavier Rival gave a talk on "MemCAD, a Modular Abstract domain for Reasoning about Memory States", at the Technical University of Vienna (TUW) on the 4th of September 2012.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence :

- Jérôme Feret, 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.
- Xavier Rival, Introduction to software verification, 12h ETD, L3, École Nationale Supérieure des Mines, Paris, France.

Master :

- Computational Biology, 9h ETD, M1. Interdisciplinary Approaches to Life Science (AIV), Master Program, Université Paris-Descartes, France.

- Patrick Cousot, Radhia Cousot, 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.

9.2.2. Supervision

PhD & HdR :

PhD: Mattéo Zanioli, Security Analysis by Abstract Interpretation [8], March 2012, Radhia Cousot (co-directed thesis with Agostino Cortesi, University of Venice), Paris VII / Università Ca' Foscari, March 12, 2012.

PhD in progress:

- Mehdi Bouaziz, Static analysis of security properties by abstract interpretation. 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, Static Analysis of Functional Programs by Abstract Interpretation, 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, Static Analysis of Functional Temporal Properties of Programs by Abstract Interpretation, November 2011, Radhia Cousot, École Normale Supérieure.

9.2.3. Juries

— Jérôme Feret reviewed the PhD thesis of Vincent Noël (Université Rennes 1, France, December 20, 2012).

— Xavier Rival reviewed the PhD theses of Arnaud Jobin (IRISA, Rennes, France, January 16, 2012), of Marie Pelleau (LINA, Nantes, France, November 29, 2012) and of Arnault Ioualalen (DALI /UPVD, Perpignan, France, December 17, 2012).

9.3. Popularization

— Xavier Rival gave a talk at the "1/2 heure de Science", at Inria Rocquencourt, on the 6th of December, 2012.

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, 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] M. ZANIOLI. *Security Analysis by Abstract Interpretation*, Université Paris VII and Università Ca'Foscari (Venice), March 2012.

Articles in International Peer-Reviewed Journals

[9] E. J. DEEDS, J. KRIVINE, J. FERET, V. DANOS, W. FONTANA. *Combinatorial complexity and compositional drift in protein interaction networks.*, in "PLoS ONE", 2012, vol. 7, n^{o} 3 [*DOI :* 10.1371/JOURNAL.PONE.0032032], http://hal.inria.fr/hal-00677889.

[10] J. FERET, T. HENZINGER, H. KOEPPL, T. PETROV. *Lumpability Abstractions of Rule-based Systems*, in "Theoretical Computer Science", 2012, vol. 431, p. 137-164 [*DOI :* 10.1016/J.TCS.2011.12.059], http://hal.inria.fr/hal-00677894.

[11] A. MINÉ. *Static analysis of run-time errors in embedded real-time parallel C programs*, in "Logical Methods in Computer Science", March 2012, vol. 8, n^{o} 1:26, 63, http://hal.inria.fr/hal-00748098.

Invited Conferences

[12] P. COUSOT. *Formal Verification by Abstract Interpretation.*, in "4th NASA Formal Methods Symposium (NFM'12)", Heidelberg, A. GOODLOE, S. PERSON (editors), LNCS, Springer-Verlag, 2012, http://www.di.ens.fr/~cousot/publications.www/Cousot-NFM2012.pdf.

[13] T. PETROV, J. FERET, H. KOEPPL. *Reconstructing Species-Based Dynamics from Reduced Stochastic Rule-Based Models.*, in "Winter Simulation Conference (WSC'12)", Berlin, Germany, C. LAROQUE, J. HIMMELSPACH, R. PASUPATHY, O. ROSE, A. M. UHRMACHER (editors), December 2012, http://hal.inria.fr/hal-00734483.

International Conferences with Proceedings

[14] M. BOUAZIZ. *TreeKs: A Functor to Make Numerical Abstract Domains Scalable*, in "4th International Workshop on Numerical and Symbolic Abstract Domains (NSAD'12)", Deauville, France, ENTCS, Elsevier, September 2012, 12, http://dx.doi.org/10.1016/j.entcs.2012.09.005.

[15] M. BOUAZIZ, F. LOGOZZO, M. FAHNDRICH. *Inference of Necessary Field Conditions with Abstract Interpretation*, in "10th Asian Conference on Programming Languages And Software (APLAS'12)", Kyoto, Japon, LNCS, December 2012, http://research.microsoft.com/apps/pubs/default.aspx?id=172534.

[16] T. CHENG, X. RIVAL. *An Abstract Domain to Infer Types over Zones in Spreadsheets*, in "19th Static Analysis Symposium (SAS'12)", Deauville, France, LNCS, September 2012, http://hal.inria.fr/hal-00760424.

[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, http://www.di.ens.fr/~cousot/publications.www/CousotCousot-POPL12-ACM-p245-258-2012.pdf.

[18] P. COUSOT, R. COUSOT, M. FAHNDRICH, F. LOGOZZO. *The Design and Implementation of a System for the Automatic Inference of Necessary Preconditions*, in "14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)", Rome, Italie, LNCS, Springer, January 2013, to appear.

[19] P. COUSOT, M. MONERAU. *Probabilistic Abstract Interpretation*, in "22nd European Symposium on Programming (ESOP'12)", Heidelberg, H. SEIDEL (editor), LNCS, Springer-Verlag, 2012, http://www.di.ens.fr/~cousot/publications.www/CousotMonerau-ESOP2012.pdf.

[20] 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.

[21] J. HAYMAN, T. HEINDEL. *Pattern Graphs and Rule-Based Models: the semantics of Kappa*, in "16th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'13)", Rome, Italy, LNCS, March 2013.

[22] A. MINÉ. *Abstract domains for bit-level machine integer and floating-point operations*, in "4th International Workshop on invariant Generation (WING'12)", Manchester, Royaume-Uni, June 2012, 16, http://hal.inria.fr/hal-00748094.

[23] A. MINÉ. *Inferring sufficient conditions with backward polyhedral under-approximations*, in "4th International Workshop on Numerical and Symbolic Abstract Domains (NSAD'12)", Deauville, France, ENTCS, Elsevier, September 2012, 12, http://hal.inria.fr/hal-00748095.

[24] A. MINÉ. *Static Analysis by Abstract Interpretation of Sequential and Multi-Thread Programs*, in "10th School of Modelling and Verifying Parallel Processes", Marseille, France, P.-A. REYNIER (editor), December 2012, http://hal.inria.fr/hal-00763076.

[25] M. PELLEAU, A. MINÉ, C. TRUCHET, F. BENHAMOU. *A constraint solver based on abstract domains*, in "14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)", Rome, Italie, LNCS, Springer, January 2013, 17, http://hal.inria.fr/hal-00748096.

[26] P. SOTIN, X. RIVAL. *Hierarchical Abstraction of Dynamic Structures*, in "10th Asian Conference on Programming Languages And Software (APLAS'12)", Kyoto, Japon, LNCS, December 2012, http://hal.inria.fr/hal-00760427.

[27] A. TOUBHANS, B.-Y. E. CHANG, X. RIVAL. *Reduced Product Combination of Abstract Domains for Shapes*, in "Verification, Model Checking and Abstract Interpretation (VMCAI'13)", Rome, Italie, LNCS, January 2013, to appear, http://hal.inria.fr/hal-00760428.

[28] 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, http://www.dsi.unive.it/~zanioli/publ/sac2012.pdf.

Books or Proceedings Editing

[29] J. FERET, A. LEVCHENKO (editors). *Proceedings of the 2nd International Workshop on Static Analysis and Systems Biology (SASB 2011)*, Electronic Notes in Theoretical Computer Science, Elsevier, June 2012, vol. 284, 137, http://hal.inria.fr/hal-00722482.

[30] A. MINÉ, D. SCHMIDT (editors). *Proceedings of the 19th International Static Analysis Symposium (SAS 2012)*, LNCS, Springer, September 2012, vol. 7460, 457, http://hal.inria.fr/hal-00748287.

References in notes

[31] 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.

[32] 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.

[33] 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, 22.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

[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, p. 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, p. 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, p. 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, p. 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, p. 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. 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.

[49] 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/.

[50] 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/.

[51] 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/.

[52] 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/.

[53] 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.

[54] A. MINÉ. *Static Analysis of Run-Time Errors in Embedded Critical Parallel C Programs*, in "ESOP'11 - 20th European Symposium on Programming", Saarbrücken, Allemagne, G. BARTHE (editor), Lecture Notes in Computer Science, Springer, 2011, vol. 6602, p. 398-418 [*DOI :* 10.1007/978-3-642-19718-5_21], http://hal.inria.fr/hal-00648038.