INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Abstraction

Abstract Interpretation

Paris - Rocquencourt

Theme : Programs, Verification and Proofs

Activity Report

2009

Table of contents

1. Team

Research Scientist

Bruno Blanchet [CR, CNRS, HdR]
Radhia Cousot [DR, CNRS, HdR]
Jérôme Feret [CR, INRIA Paris–Rocquencourt]
Antoine Miné [CR, CNRS]
Xavier Rival [CR, INRIA Paris–Rocquencourt]

Faculty Member

Patrick Cousot [Team leader, Professor/Professeur, ENS, HdR]
Laurent Mauborgne [Assistant Professor/Maître de conférences, ENS, HdR]

Technical Staff

Élodie-Jane Sims [Research engineer, ENS, — Sep. 2009, INRIA, Oct. 2009 —]

PhD Student

David Cadé [Sep. 2009 —]
Liqian Chen [Mar. 2009 —]
Pietro Ferrara [— Mar. 2009]
Vincent Laviron [Sep. 2009 —]
Jérémy Leconte [Sep. 2009 —]
Matteo Zanioli [Sep. 2009 —]

Post-Doctoral Fellow

Julien Bertrane
Axel Simon
Colas Le Guernic [Nov. 2009 —]

Visiting Scientist

Miriam Paiola [Università di Padova, Sep. 2009 — Apr. 2010]
Ben Smyth [University of Birmingham, Sep. 2009 — Feb. 2010]

Administrative Assistant

Joëlle Isnard [Administrative Head DI, ENS]
Elisabeth Baque [INRIA, — Aug. 2009]
Emmanuelle Grousset [INRIA, Aug. 2009 — Nov. 2009]
Nathalie Abiola [INRIA, Nov. 2009 —]

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 and development 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

In February 2009, the industrialization of the ASTRÉE analyzer started with the signature of a license agreement with AbsInt Angewandte Informatik GmbH. ASTRÉE should be commercially available from AbsInt in the near future.

In 2009, the Space Software Validation using Abstract Interpretation ESA-ITI project with ESA, CEA, and Astrium Space Transportation concluded that the new generation of static analyzers, including ASTRÉE, can be used to improve the safety of embedded critical space software (in the past, industrial codes analyzed by ASTRÉE originated mainly from the avionic industry).

3. Scientific Foundations

3.1. Abstract Interpretation Theory

The abstract interpretation theory [6], [77], [81] 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.

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...).

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. [80], [12], [15]) 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 [82]. 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 thousands of ideas, concepts, reasonings, methods, and tools on formal program analysis and verification [6], [81];
- guiding the correct formal design of *abstract semantics* [12], [15] and automatic tools for *program analysis* (computing an abstract semantics) and *program verification* (proving that an abstract semantics satisfies an abstract specification) [78].

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 5.1), 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 is used in the aerospace industry to verify very large, synchronous, time-triggered, real-time, safety-critical, embedded software.

3.3. Advanced Introductions to Abstract Interpretation

The informal presentation "Abstract Interpretation in a Nutshell" aims at providing a short intuitive introduction to the theory. A more comprehensive introduction to abstract interpretation is available online[1]. The paper entitled "Basic concepts of abstract interpretation" [79] and an elementary "course on abstract interpretation"[2] can also be found on the web.

4. Application Domains

4.1. Certification of Safety Critical Software

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

This development of safe and secure critical software requires formal methods so as to ensure that they do not go wrong, and will behave as specified. In particular, testing or bug finding methods 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.

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. 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.

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

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 [76], [83], 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. Security Protocols

Security protocols use cryptography in order to guarantee the security of exchanges over an insecure network, such as the Internet. The design of security protocols is notoriously error-prone: errors have been found in many published protocols. Security errors can have serious consequences, such as loss of money in the case of electronic commerce. Moreover, security errors cannot be detected by testing, because they appear only in the presence of a malicious adversary. Security protocols are therefore an important area for formal verification.

The work of the ABSTRACTION project-team on security protocols has led to the development of two successful automatic protocol verifiers, PROVERIF in the formal model and CRYPTOVERIF in the computational model, and we plan to pursue research on this topic, in particular with extensions to CRYPTOVERIF.

4.3. Abstraction of Biological Cell Signaling Networks

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

Contextual graph-rewriting systems allow a concise description of these networks, which leads to a scalable method for modeling them. Then abstract interpretation allows the abstraction of these systems properties. First qualitative abstractions (such as over approximation of complexes that can be built) provide both debugging information in the design phases (of models) and static information that are 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 abstraction to quantitative abstraction.

5. Software

5.1. The Astrée Static Analyzer

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

The ASTRÉE static analyzer [76], [83] 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 and 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);
- 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 should be commercially available in December 2009.

5.2. The Apron Numerical Abstract Domain Library

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

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

The APRON library is not tied to a particular numerical abstraction but instead provides several domains with various precision versus cost trade-offs (including intervals, octagons, linear equalities and polyhedra). A specific C API was designed for domain developers to minimize the effort when incorporating a new abstract domain: only few domain-specific functions need to be implemented while the library provides various generic services and fallback methods (such as scalar and interval operations for most numerical data-types, parametric reduced products, and generic transfer functions for non-linear assignments). For the analysis designer, the APRON library exposes a higher-level API with C, C++, and OCaml 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 under the LGPL license. 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 86 000 lines of C, C++, and OCaml. This year has seen the release of version 0.9.10, which mainly includes bugfixes and minor improvements in the API and build system. We also published a tool paper [25] describing the library and its design philosophy.

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

5.3. Translation Validation

Participant: Xavier Rival [correspondent].

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

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

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

5.4. ProVerif

Participants: Bruno Blanchet [correspondent], Xavier Allamigeon [April–July 2004], Ben Smyth [Sept. 2009–Feb. 2010].

PROVERIF (www.proverif.ens.fr) is an automatic security protocol verifier, in the formal model (so called Dolev–Yao model). In this model, cryptographic primitives are considered as black boxes. This protocol verifier is based on an abstract representation of the protocol by Horn clauses. Its main features are:

- It can handle many different cryptographic primitives, including shared- and public-key cryptography (encryption and signatures), hash functions, and Diffie–Hellman key agreements, specified both as rewrite rules or as equations.
- It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. This result has been obtained thanks to some well-chosen approximations. This means the verifier can give false attacks, but if it claims that the protocol satisfies some property, then the property is actually satisfied. PROVERIF also provides attack reconstruction: when it cannot prove a property, it tries to reconstruct an attack, that is, an execution trace of the protocol that falsifies the desired property.

The PROVERIF verifier can prove the following properties:

- secrecy (the adversary cannot obtain the secret);
- authentication and more generally correspondence properties, of the form “if an event has been executed, then other events have been executed as well”;
- strong secrecy (the adversary does not see the difference when the value of the secret changes);
- equivalences between processes that differ only by terms;

PROVERIF has been used by researchers for studying various kinds of protocols, including electronic voting protocols, certified email protocols, and zero-knowledge protocols. It has been used as a back-end for the tool TULAFALE implemented at Microsoft Research Cambridge, which verifies web services protocols. It has also been used as a back-end for verifying implementations of protocols in F# (a dialect of ML included in .NET), by Microsoft Research Cambridge and the joint INRIA-Microsoft research center.

PROVERIF is freely available on the web, at www.proverif.ens.fr, under the GPL license.

5.5. CryptoVerif

Participant: Bruno Blanchet [correspondent].

CRYPTOVERIF (www.cryptoverif.ens.fr) is an automatic protocol prover sound in the computational model. In this model, messages are bitstrings and the adversary is a polynomial-time probabilistic Turing machine. CRYPTOVERIF can prove:

- secrecy [73];
- correspondences [72], which include in particular authentication.

CRYPTOVERIF provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions.

The generated proofs are proofs by sequences of games, as used by cryptographers. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary. CRYPTOVERIF can also evaluate the probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions (exact security).

CRYPTOVERIF is still at a rather early stage of development, but it has already been used for a study of Kerberos in the computational model. It is also used as a back-end for verifying implementations of protocols in F# at Microsoft Research Cambridge and at the joint INRIA-Microsoft research center.

CRYPTOVERIF is freely available on the web, at www.cryptoverif.ens.fr, under the CeCILL license.

6. New Results

6.1. Space Software Validation using Abstract Interpretation

Participants: Olivier Bouissou [CEA LIST], Éric Conquet [ESA - ESTEC], Patrick Cousot, Radhia Cousot, Jérôme Feret, Khalil Ghorbal [CEA LIST], Éric Goubault [CEA LIST], David Lesens [Astrium ST], Laurent Mauborgne, Antoine Miné, Sylvie Putot [CEA LIST], Xavier Rival, Michel Turin [GTI6].

In [20], we report the results of the ESA funded project (see 7.2) on the use of abstract interpretation to validate critical real-time embedded space software. Abstract interpretation is industrially used since several years, especially for the validation of the Ariane 5 launcher. However, the limitations of the tools used so far prevented a wider deployment. Astrium Space Transportation, CEA, and ENS have analyzed the performances of two recent tools on a case study extracted from the safety software of the ATV:

- ASTRÉE, developed by ENS and CNRS, to check for run-time errors,
- FLUCTUAT, developed by CEA, to analyze the accuracy of numerical computations.

The conclusion of the study is that the performance of this new generation of tools has dramatically increased (no false alarms and fine analysis of numerical precision).

Thanks to a case study representative of the software developed at Astrium ST, the results of this study will be applicable to any type of embedded critical real-time space software (launchers, satellites, spacecrafts, and space probes) developed in C. They will improve the quality of software (fewer residual bugs) and will at the same time dramatically decrease the costs of robustness testing. The study has also hinted towards some directions of improvement for the tools. As a conclusion, the Technology Readiness Level (TRL) for ASTRÉE and FLUCTUAT on space software is evaluated between 4 (component and/or breadboard validation in laboratory environment) and 5 (component and/or breadboard validation in relevant environment).

6.2. Why does Astrée scale up?

Participants: Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival.

ASTRÉE was the first static analyzer able to prove automatically the total absence of runtime errors of actual industrial programs of hundreds of thousand lines. What makes ASTRÉE such an innovative tool is its scalability, while retaining the required precision, when it is used to analyze a specific class of programs: that of reactive control-command software.

In [14], we discuss the important choice of algorithms and data-structures we made to achieve this goal. However, what really made this task possible was the ability to also take semantic decisions, without compromising soundness, thanks to the abstract interpretation framework. We discuss the way the precision of the semantics was tuned in ASTRÉE in order to scale up, the differences with some more academic approaches and some of the dead-ends we explored. In particular, we show a development process which was not specific to the particular usage ASTRÉE was built for, hoping that it might prove helpful in building other scalable static analyzers.

6.3. Design of the Apron Library

Participants: Bertrand Jeannet [INRIA Rhône-Alpes], Antoine Miné.

In [25], we describe APRON, a freely available library dedicated to the static analysis of the numerical variables of programs by abstract interpretation (5.2). Its goal is to provide analysis implementers with reference implementations of classic domains, encourage the research in new numerical abstract domains, and provide a teaching and demonstration tool to disseminate knowledge on abstract interpretation. The tool paper describes the different components of APRON and, more importantly, its unique design philosophy. The core idea is that the API corresponds to a concrete semantics that domains are free to approximate in any sound way, thus enabling efficient abstract algorithms and incremental development with a precise control over trade-offs between semantic precision, algorithmic cost, and human cost of domain development. This is in contrast to other libraries that hardcode an abstract semantics into their API, which then becomes tied to particular abstraction choices. Our concrete semantics is rich, allowing to express in particular non-linear and floating-point computations, but we provide default abstractions for these complex cases. Indeed, APRON aims at conciliating two kinds of users with conflicting requirements: analysis designers wishing for a rich, high-level, and domain-independent API, and domain designers, wishing to implement a minimal set of low-level entry-points.

6.4. Hierarchy of Semantics by Abstract Interpretation and Formal Proofs

Participant: Jeremy Leconte.

Using the Coq proof assistant, we define a maximal trace semantics describing terminating and diverging computations of a simple functional language. This trace semantics is abstracted into a relational semantics which is in turn abstracted into a reduction semantics. The soundness of each abstraction is proved along with the equivalence between small steps and big steps of computation. An originality of the semantics is that it defines finite behaviors inductively and infinite behaviors coinductively while it avoids the duplication of rules common to the two behaviors. In consequence the semantics definition is simplified as well as some proofs that uses the definition.

This work is summarized in [61].

6.5. Semantics Design by Abstract Interpretation

The *semantics of a programming language* defines the semantics of its programs. The *semantics of a program* formally defines all its possible executions in all possible execution environments. The theory of abstract interpretation can be used to formally design an abstract semantics from a more concrete one (or inversely) which yields different styles of semantics such as operational, relational, denotational, axiomatic semantics, etc. Each such semantics describes program executions at different levels of abstraction. The latest results are related abstract semantics of the lambda-calculus [12] and of resolution-based programming languages [15].

6.5.1. Bi-inductive Definitions and Bifinitary Semantics of the Eager Lambda-Calculus

Keywords: *bi-inductive definition*, *big-step semantics*, *divergence*, *inductive definition*, *natural semantics*, *operational semantics*, *relational semantics*, *small-step semantics*, *structural semantics*.

Participants: Patrick Cousot, Radhia Cousot.

We have introduced an order-theoretic generalization of set-theoretic inductive definitions. This generalization covers inductive, co-inductive, and bi-inductive definitions, including non-monotonic ones, and is preserved by abstraction. This allows the structural operational semantics to describe simultaneously the finite/terminating and infinite/diverging behaviors of programs. This is illustrated on the structural bifinitary semantics of the call-by-value λ-calculus at various levels of abstraction including small/big-step trace/relational/operational semantics [12].

6.5.2. Abstract Semantics of Resolution-Based Logic Languages

Keywords: *Herbrand semantics*, *abstract semantics*, *bottom-up semantics*, *logic programming*, *parsing*, *s-semantics*, *top-down semantics*.

Participants: Patrick Cousot, Radhia Cousot, Roberto Giacobazzi [Universita' degli Studi di Verona].

The abstract interpretation point of view on context-free grammars [80] has been extended to resolution-based logic programs and proof systems in [15]. Starting from a transition-based small-step operational semantics of PROLOG-like programs (akin to the Warren Machine), we consider maximal infinite derivations for the transition system from most general goals. This semantics is abstracted by instantiation to terms and furthermore to ground terms, following the so called c and s-semantics approach. Orthogonally, these sets of derivations can be abstracted to SLD-trees, call patterns and models, as well as interpreters providing effective implementations (such as PROLOG or lazy PROLOG). These semantics can be presented in bottom-up fixpoint form. This abstract interpretation-based construction leads to classical bottom-up semantics (such as the s-semantics of computed answers of Giorgio Levi, the c-semantics of correct answers of Keith Clark, and the minimal-model semantics of logical consequences of Maarten van Emden and Robert Kowalski). The approach is general and can be applied to infinite and top-down semantics.

6.6. Verification of Security Protocols in the Formal Model

The formal model of protocols, or Dolev–Yao model is an abstract model in which messages are represented by terms. Our protocol verifier PROVERIF relies on this model. This year, we have written a book chapter on the resolution algorithm at the heart of PROVERIF and have implemented several extensions of PROVERIF.

6.6.1. Book Chapter on Using Horn Clauses for the Verification of Security Protocols

Keywords: *Horn clauses*, *automatic verification*, *resolution*, *security protocols*.

Participant: Bruno Blanchet.

We have written a book chapter [32] that introduces the theory behind the protocol verifier PROVERIF. It explains the abstract representation of protocols by Horn clauses used by PROVERIF. It also presents and proves correct the resolution algorithm that is used to determine whether a fact is derivable from these clauses. From this information, security properties of the protocol can be inferred; we focus on secrecy in this chapter, but this method can also prove other security properties, including authentication and process equivalences.

6.6.2. Extensions of ProVerif

Keywords: *attack reconstruction*, *automatic verification*, *documentation*, *interface*, *security protocols*.

Participants: Bruno Blanchet, Ben Smyth.

In the frame of a contract with CELAR (see Section 7.4), we have implemented several extensions of PROVERIF.

We have implemented the reconstruction of attacks against injective correspondences. Injective correspondences are properties of the form: for each execution of a certain event e_1, there is a distinct execution of another event e_2. PROVERIF could already reconstruct attacks in which e_1 is executed without e_2 being executed (which also contradict a non-injective correspondence); we have extended it to reconstruct attacks in which e_1 is executed twice and e_2 is executed only once.

We have also improved the interface of PROVERIF. We have implemented a new front-end with types, parametric processes, function macros, and specific constructions for representing tables of keys. The goal of this new front-end is to make it easier for users to model protocols, and to detect some bugs in protocol specifications (in particular thanks to typing). This input language is also closer to the one of CRYPTOVERIF, so that many examples of protocols can be given to both tools with no or little modification.

Finally, we are currently writing a more detailed documentation of PROVERIF, including a tutorial with examples of protocols, to facilitate its access to users who are not experts in the pi-calculus or formal methods.

6.7. Verification of Security Protocols in the Computational Model

The computational model of protocols considers messages as bitstrings, which is more realistic than the formal model, but also makes the proofs more difficult. Our verifier CRYPTOVERIF is sound in this model. This year, we have continued our case study of Kerberos, have built a compiler from CRYPTOVERIF specifications to implementations of protocols, and have implemented extensions of CRYPTOVERIF.

6.7.1. Computationally Sound Mechanized Proofs for Basic and Public-Key Kerberos

Keywords: *Kerberos*, *automatic verification*, *computational model*, *key usability*, *security protocols*.

Participants: Bruno Blanchet, Aaron Jaggard [Rutgers University], Jesse Rao, Andre Scedrov [University of Pennsylvania], Joe-Kai Tsay [Ruhr-University Bochum].

We have extended our computationally sound analysis of Kerberos 5 with CRYPTOVERIF [74]. In particular, we have extended our definition of key usability from IND-CCA2 key usability to INT-CTXT (ciphertext integrity) key usability, and we have proved that the session keys of Kerberos satisfy this new definition. This work was presented in [28].

6.7.2. Automatic Translation from CryptoVerif Specifications to Implementations

Keywords: *automatic verification*, *compilation*, *computational model*, *implementations*, *security protocols*.

Participant: David Cadé.

We have implemented a compiler [44][29] that takes a CryptoVerif specification and generates an implementation of the protocol in OCaml. The goal of this work is to obtain implementations of security protocols proved secure in the computational model.

In future works, we will prove that our compiler is correct, that is, the semantics of the generated code corresponds to the semantics of the specification. Therefore, if CryptoVerif can prove a property on the protocol, then the implementation will also satisfy this property.

6.7.3. Extensions of CryptoVerif

Keywords: *automatic verification*, *computational model*, *security protocols*.

Participant: Bruno Blanchet.

We have extended CRYPTOVERIF in order to handle the computational Diffie–Hellman assumption, which allows us to prove a signed Diffie–Hellman protocol in the random oracle model, for instance. We allow manually guided elimination of collisions between random numbers; this extension is particularly helpful for passwords: passwords have a non-negligible probability of being guessed, so the automatic elimination of collisions for passwords often leads to unacceptably large probabilities of attacks. We also allow the manual insertion of events, whose probability of execution is later bounded by CRYPTOVERIF. Thanks to these extensions, we plan to prove the AuthA password-based key exchange protocol (a variant of EKE, Encrypted Key Exchange).

6.8. Verification of Security Protocols: Formal Model and Computational Model

Participants: Martín Abadi [Microsoft Research, Silicon Valley and University of California, Santa Cruz], Bruno Blanchet, Hubert Comon-Lundh [INRIA, ENS Cachan, and RCIS, AIST].

In [19], we discuss progress in the verification of security protocols. Focusing on a small, classic example, the Wide-Mouth Frog protocol, we stress the use of program-like representations of protocols, and their automatic analysis in symbolic and computational models. Specifically, we compare two analyses of this protocol. The first one relies on PROVERIF for verifying the protocol in the symbolic model and uses a recent computational soundness theorem in order to infer security in the computational model. The second one uses CRYPTOVERIF to obtain a direct proof in the computational model.

6.9. Analysis of Biological Pathways

We have introduced a framework to design and analyze biological networks. We focus 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).

6.9.1. Automatic Reduction of Differential Semantics

Keywords: *biology*, *differential semantics*, *protein-protein interaction networks*, *verification*.

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

We have developed an abstract interpretation-based framework that enables the computation of scalable differential semantics for protein-protein interaction networks. This framework uses indistinguishably techniques in order to detect and prove that some potential correlations between the states of some distinct parts in protein species have no impact on the dynamic of the networks. These information drive the computation of an abstract differential system over a set of self-consistent abstract observables. Results are sound since trajectories in the abstract system are projections of the trajectories in the concrete system. This framework gives new insights in order to describe evolution between systems: indeed several networks can be compared according to the relative amount of control between protein-protein interactions.

This framework has been published in [17] and presented in [51], [50], and [53].

6.9.2. Automatic Reduction of Stochastic Semantics

Keywords: *biology*, *protein-protein interaction networks*, *stochastic semantics*, *verification*.

Participants: Jérôme Feret, Heinz Koeppl [École Polytechnique Fédérale de Lausanne], Tatjana Petrov [École Polytechnique Fédérale de Lausanne].

We have proposed an abstract interpretation-based framework for reducing the state-space of stochastic semantics for protein-protein interaction networks. Our approach consists in quotienting the state-space of networks. Yet interestingly, we do not apply the widely-used strong lumpability criterion which imposes that two equivalent states behave similarly with respect to the quotient, but a weak version of it. More precisely, our framework detects and proves some invariants about the dynamics of the system: indeed the quotient of the state-space is such that the probability of being in a given state knowing that this state is in a given equivalence class, is an invariant of the semantics.

Then we have introduced an individual-based stochastic semantics (where each agent is identified by a unique identifier) for the programs of a rule-based language and we use our abstraction framework to derive a sound population-based semantics and a sound fragments-based semantics, which give the distribution of the traces respectively for the number of instances of molecular species and for the number of instances of partially defined molecular species. These partially defined species are chosen automatically thanks to a dependency analysis which is also described in the paper.

Interestingly, we have showed that the criteria that we were using in [17] in order to abstract the differential semantics were not sound regarding to the stochastic semantics. Indeed, we had to strengthen these criteria in order to respect the stochastic semantics. As a consequence the reduction factor is far less impressive in the case of the stochastic semantics. This reflects the fact that the stochastic semantics is a much more expressive and interesting object than the differential semantics and also that it is much more difficult to abstract.

This work has been presented in [55] and [56].

6.9.3. Rule Refinements

Keywords: *biology*, *concurrency*, *protein-protein interaction networks*, *refinements*.

Participants: Vincent Danos [University of Edinburgh], Jérôme Feret, Russel Harmer [Paris VII], Jean Krivine [Harvard Medical School], Elaine Murphy [University of Edinburgh].

We have proposed a formal framework to refine rule-based protein-protein interaction networks while preserving their stochastic and their differential semantics. Refinements is a key process in rule-based modeling. Refining an interaction allows tuning the kinetics of an interaction according to some constraints in the context of the interacting proteins.

In [84], we had proposed a framework to make homogeneous refinements. In such a homogeneous refinement, the accuracy of the refinement is the same for each protein of a given type. In [34], we have extended this framework in order to make heterogeneous refinements, where each agent in a given pattern can be refined independently.

6.9.4. *Investigation of a Biological Repair Scheme*

Keywords: *biology*, *protein-protein interaction networks*, *refinements*.

Participants: Vincent Danos [University of Edinburgh], Jérôme Feret, Walter Fontana.

In [23], we investigated an interaction pattern for the allocation of a scarce biological resource where and when it is needed. It is entirely based on a mass action stochastic dynamics. Domain-domain binding plays a crucial role in the design of the pattern which we therefore present using a rule-based approach where binding is an explicit primitive. We also use a series of refinements, starting from a very simple interaction set, which we feel gives an interesting and intuitive rationale for the working of the final repair scheme.

6.9.5. *Meta-Language*

Keywords: *biology*, *concurrency*, *protein-protein interaction networks*, *refinements*.

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

Rule-based modeling has already proved to be successful for taming the combinatorial complexity, typical of cellular signaling networks, caused by the combination of physical protein-protein interactions and modifications that generate astronomical numbers of distinct molecular species. However, traditional rule-based approaches, based on an unstructured space of agents and rules, remain susceptible to other combinatorial explosions caused by mutated and/or splice variant agents, that share most but not all of their rules with their wild-type counterparts; and by drugs, which must be clearly distinguished from physiological ligands.

In [16], we define a syntactic extension of Kappa, an established rule-based modeling platform, that enables the expression of a structured space of agents and rules that allows us to express mutated agents, splice variants, families of related proteins and ligand/drug interventions uniformly. This also enables a mode of model construction where, starting from the current consensus model, we attempt to reproduce in numero the mutational—and more generally the ligand/drug perturbational—analyses that were used in the process of inferring those pathways in the first place.

6.10. Shape analysis of a realistic memory model

Participant: Vincent Laviron.

In [60], we have refined the shape analyzer Xisa so as to use a new memory model featuring pointer arithmetic and union types.

6.11. Checkmate: a Generic Static Analyzer of Java Multithreaded Programs

Participant: Pietro Ferrara.

In [24], we present Checkmate, a generic static analyzer of Java multithreaded programs based on the abstract interpretation theory. It supports all the most relevant features of Java multithreading, as dynamic unbounded thread creation, runtime creation of monitors, and dynamic allocation of shared memory. We implement a wide set of properties, from the ones interesting also for sequential programs, e.g. division by zero, to the ones typical of multithtreaded programs, e.g. data races. We analyze several external case studies and benchmarks with Checkmate, and we study the experimental results both in term of precision and efficiency. It turns out that the analysis is particularly accurate and we are in position to analyze programs composed by some thousands of statements and a potentially infinite number of threads. As far as we know, Checkmate is the first generic static analyzer of Java multithreaded programs.

6.12. Static Analysis via Abstract Interpretation of Multithreaded Programs

Participant: Pietro Ferrara.

Pietro Ferrara has written his PhD on static analysis of multithreaded programs via abstract interpretation. In this PhD, the design of a generic analyzer for multithreaded programs is presented. First of all, the happens-before memory model (as an over-approximation of the Java memory model) in fixpoint form is defined and we abstract it with a computable semantics. It is shown how to design a computable abstract semantics, and the correctness of the resulting analysis is proved in a formal way. Then we define and abstract a new property focused on the non-deterministic behaviors due to multithreading, e.g. the arbitrary interleaving during the execution of different threads. Different levels of determinism are defined, relating this property to the presence of data races. This theoretical framework is applied to Java. In particular, a concrete semantics of bytecode language is defined following its specification. Then it is abstracted in order to track the information required by the analysis of multithreaded programs. The core is an alias analysis that approximates references in order to identify threads, to check the accesses to the shared memory, and to detect when two threads own a common monitor thereby inferring which parts of the code cannot be executed in parallel. The generic analyzer described above has been fully implemented, leading to Checkmate, the first generic analyzer of Java multithreaded programs. Some experimental results are reported and deeply studied. An additional contribution is about the extension of an existing industrial generic analyzer, Clousot, to the checking of buffer overrun. It turns out that this analysis is scalable and precise. In summary, we present an application of an existing, industrial, and generic static analyzer to a property of practical interest, showing the strength of this approach in order to develop useful tools for developers.

Pietro Ferrara's report [11] summarizes his work in this topic. He defended his PhD on May 22, 2009.

6.13. SubPolyhedra: A (more) scalable approach to infer linear inequalities

Participants: Vincent Laviron, Francesco Logozzo [Microsoft Research, Redmond].

In [27], we introduce Subpolyhedra (SubPoly) a new numerical abstract domain to infer and propagate linear inequalities. SubPoly is as expressive as Polyhedra, but it drops some of the deductive power to achieve scalability. SubPoly is based on the insight that the reduced product of linear equalities and intervals produces powerful yet scalable analyses.

Precision can be recovered using hints. Hints can be automatically generated or provided by the user in the form of annotations.

We implemented SubPoly on the top of Clousot, a generic abstract interpreter for .Net. Clousot with SubPoly analyzes very large and complex code bases in few minutes. SubPoly can effciently capture linear inequalities among hundreds of variables, a result well-beyond state-of-the-art implementations of Polyhedra.

6.14. Refining Abstract Interpretation-based Static Analyses with Hints

Participants: Vincent Laviron, Francesco Logozzo [Microsoft Research, Redmond].

The existing approaches for refining static analyses focus on the refinement either of the elements of abstract domains or of the transfer functions.

In [26], we focus our attention on the loss of precision induced by abstract domain operations. We introduce a new technique, hints, which allows to systematically refine the operations defined over elements of an abstract domain. We formally define hints in the abstract interpretation theory, we prove their soundness, and we characterize two families of hints: syntactic and semantic.

6.15. Approximated Inference of Linear Invariants

Participant: Axel Simon.

The Two-Variable-Per-Inequality abstract domain can be used to infer linear invariants of the form $ax_i + bx_j \leq c$ where $a, b, c \in \mathbb{N}$ and x_i, x_j are two (program) variables. The domain is more efficient than the classic polyhedra domain and more expressive than the Octagon domain. A journal version [18] of the previous conference paper [85] has been accepted pending minor revisions.

6.16. An Abstract Domain to Infer Interval Linear Relationships

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

In previous work [75], we proposed a sound floating-point version of the polyhedron abstract domain. Soundness was achieved despite rounding errors by leveraging previous works on rigorous linear programming and designing a version of Fourier–Motzkin elimination using interval arithmetics internally.

In [21], we propose an alternate construction where intervals appear explicitly in the abstract representation. Hence, the domain, so called *interval polyhedra*, can represent conjunctions of constraints of the form $\Sigma_k[a_k; b_k]x_k \leq c$. Thus, we avoid the loss of precision that occurred in our previous work when converting the interval constraints that appeared naturally into scalar ones at the end of each operation. An added benefit is that this domain is strictly more expressive than regular polyhedra, as it can express some non-convex and even unconnected sets. The operations are based on interval linear programming and an interval variant of Fourier–Motzkin elimination, and can be implemented soundly using only floating-point arithmetics, thus ensuring a good time and memory complexity (in particular, we are free of the issue of coefficient explosion occurring in classic implementations that employ arbitrary precision exact rationals). Our prototype implementation shows encouraging preliminary implementation results. In particular, it can prove some disjunctive and non-linear properties out of the scope of the classic polyhedra domain.

7. Contracts and Grants with Industry

7.1. ES_PASS Contract

ES_PASS (Embedded Software Product-based ASSurance) is an ITEA European project grouping technology and tool providers as well as industrial end-users in the field of embedded software for automotive, avionic, railway and space transportation (AbsInt Angewandte Informatik GmbH, Airbus France, CEA/LIST, CS Systèmes d'Information, DaimlerChrysler AG, EADS Astrium SAS, EADS Innovation Works, École Normale Supérieure (ENS), Esterel technologies, FéRIA (IRIT & ONERA), Fraunhofer FIRST, Institut für Bahntechnik (IFB), Saarland University, Siemens VDO, Technical University Munich, Technical University of Madrid, Thales Avionics, Thales Transport). The objective of the participation of the ABSTRACTION project-team to ES_PASS is to confront the ASTRÉE analyzer to a wide range of industrial applications in order to evaluate its practical applicability and prepare its industrialization. Patrick Cousot is the principal investigator for this action.

7.2. SSVAI Contract

SSVAI (Space Software Validation using Abstract Interpretation) is an ESA-ITI project (European Space Agency's Innovative Triangle Initiative) with Astrium Space Transportation, the CEA, the ENS, and the École polytechnique. The activity of the ABSTRACTION project-team in this project is mainly to apply the ASTRÉE static analyzer to the MSU (Monitoring Software Unit) code of the ATV (Automated Transfer Vehicle) for the ISS (International Space Station).

Upon completion of the project, we successfully analyzed several versions of a Scade model of the MSU controller compiled into C (including versions generated by different Scade compilers, and using different generation options). The study demonstrated the ability of ASTRÉE to handle Scade-generated code. It showed that, although the library of abstract domains built in ASTRÉE from our experience on avionics software is sufficient in some cases, achieving zero false alarms would require the development of new abstract domains adapted to the aspects of control theory specific to space control. However, it also showed that this could be mitigated by introducing a few numerical limiters at strategic locations in the code. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

7.3. Asbaprod Contract

ASBAPROD (ASsurance BAsée PRODduit) is an industrial project on static program analysis by abstract interpretation with Airbus France which objective is determined annually.

The main work in 2009 consisted in:

- designing a generic frontend for our translation validator (see 5.3); this way, end-users can implement new frontends and use them together with the translation validator in order to certify the compilation of safety critical software written in arbitrary languages (the prover and abstraction phases are common to the generic interface and the C version of the translation validator);
- providing a manual for the translation validator (see 5.3);
- designing a backward analyzer so as to help alarms diagnostic; in the case the forward analysis raises an alarm, the backward analysis computes automatically a refined set of invariants, which characterize better the set of program executions that may cause an error at the alarm point; in some cases it is even possible to prove that set empty (*i.e.*, that the alarm was in fact a false alarm);
- enriching the ASTRÉE Analyzer (see 5.1), so that all floating-points errors due to uninitialized floats can be detected, especially those that are due to *NaN* or infinities coming from unitialized or volatile (sensors) values.

Patrick Cousot is the principal investigator for this action.

7.4. Contract with CELAR

In the frame of a contract with CELAR (*Centre d'Électronique de l'Armement*), we implement extensions of PROVERIF in order to make it easier to use: extensions to attack reconstruction, improvements to the user interface and to the documentation. Bruno Blanchet is the principal investigator for this action.

7.5. Survol project

SURVOL is an FNRAE (*Fondation de Recherche pour l'Aéronautique et l'Espace*) project (2008–2011) with the University Paul Sabatier Toulouse III and the ENS. The objective is to study automatic verification techniques for embedded robust and stable control software in aerospace still functioning in degraded mode and based on non-smooth optimization and linear matrix inequality (LMI) programming. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

7.6. Ascert project

ASCERT is an FNRAE (*Fondation de Recherche pour l'Aéronautique et l'Espace*) project (2009–2012) between the INRIA-Bretagne Atlantique, the INRIA Rhône-Alpes, the INRIA Paris-Rocquencourt, and the ENS. The goal of this project is to study the use of theorem provers so as to formally certify static analyses. Several approaches are considered: the certification of the static analyzer (once and for all) versus the independent certification of each analyzer run. The project will focus on the certification of some parts of the APRON library and the ASTRÉE analyzer by means of the COQ theorem prover. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

7.7. Sardanes project

SARDANES is an FNRAE (*Fondation de Recherche pour l'Aéronautique et l'Espace*) project (2009–2012) between the University of Perpignan and the ENS. 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.

7.8. Astrée exploitation license agreement

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 should be commercially available from AbsInt in December 2009. Patrick Cousot is the scientific contact.

8. Other Grants and Activities

8.1. Controvert ANR

The CONTROVERT project (2005–2008) brings together control-theory researchers of ONERA/DCSD and the Université Paul Sabatier of Toulouse and computer scientists from the ABSTRACTION project-team. A first objective is to bridge the gap between control-theory-based methods for analyzing properties of models of systems and their controllers (e.g. robustness) by continuous Lagrangian overapproximation of the system trajectories and abstract-interpretation-based methods for analyzing control/command programs (e.g. safety properties) in open loop. A second objective is to use the results of the control-command theoretic analysis of the closed loop to support the program analysis in the context of the controlled system. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

8.2. FormaCrypt ARA

The ABSTRACTION project-team coordinates the FORMACRYPT project, on "formal proofs and probabilistic semantics in cryptography" (project web site: http://www.di.ens.fr/~blanchet/formacrypt/index.html). This project is financed by the *Agence Nationale pour la Recherche*, in the frame of the *Action de Recherche Amont Sécurité, Systèmes embarqués et Intelligence Ambiante (ARA SSIA)*. This project of a duration of 3 years and half (January 2006–July 2009) brings together researchers from the INRIA project-teams ABSTRACTION and CASCADE (LIENS, *Laboratoire d'Informatique de l'École Normale Supérieure*), SECSI (LSV, *Laboratoire Spécification et Vérification, ENS Cachan*), and CASSIS (LORIA, *Laboratoire Lorrain de Recherche en Informatique et ses Applications*), as well as Martín Abadi as scientific advisor. The goal of this project is to bridge the gap between the formal and computational models of security protocols, so as to obtain automatic proofs of protocols valid in the computational model. This project has led to 39 publications in international conferences, workshops, and journals, to the implementation of two tools, CRYPTOVERIF and an extension of AVISPA, and strongly contributed to the organization of a series of international workshops (the workshops on Formal and Computational Cryptography, FCC). Bruno Blanchet is the principal investigator for this action.

8.3. Thésée ANR

The objective of the THÉSÉE project (2006–2009) is to develop static analysis techniques for proving the absence of runtime errors in asynchronous (real-time) programs. The project is in cooperation with EDF and Airbus France. The main problem is to scale up traditional sequential static analysis methods so as to cope with the combinatorial explosion resulting form the interleaving of communications and interactions through shared variables in a parallel execution of the asynchronous processes. Patrick Cousot and Radhia Cousot are the principal investigators for this action.

8.4. AbstractCell ANR

The project ABSTRACTCELL (2009–2013) is sponsored by the ANR-Chair of Excellence program 2009. 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.

9. Dissemination

9.1. Interaction with the Scientific Community

9.1.1. Academy Members, Professional Societies

Patrick Cousot is a member of the Academia Europaea.

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

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

9.1.2. Collective Responsibilities

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

Radhia Cousot was head of the Abstract Interpretation group at École Polytechnique under a convention between the CNRS, the École Normale Supérieure and the École Polytechnique until April 1st, 2009.

Laurent Mauborgne is assistant director of studies in computer science at ENS and member of the *commission de spécialistes* (hiring committee) of ENS and VERIMAG laboratory in Grenoble.

Patrick Cousot, Laurent Mauborgne, Xavier Rival, Élodie-Jane Sims, and Antoine Miné are members of the lab council of the Laboratoire d'Informatique de l'École Normale Supérieure.

9.1.3. Editorial Boards and Program Committees

Bruno Blanchet is associate editor of the International Journal of Applied Cryptography (IJACT).

Bruno Blanchet was member of the program committee of the IEEE Computer Security Foundations Symposium (CSF 2009), the Workshop on Formal and Computational Cryptography (FCC 2009), the 13th Annual Asian Computing Science Conference (ASIAN 2009), and the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010).

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 10th International Conference on Verification, Model-Checking and Abstract Interpretation (VMCAI'09), the 12th International Conference on Hybrid Systems: Computation and Control (HSCC'09), the Structural Operational Semantics (SOS'09) conference, the 14th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'09), the 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM'09), the 11th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2010), and the 19th European Symposium on Programming (ESOP'10).

Radhia Cousot is member of the advisory board of the Higher-Order Symbolic Computation journal (HOSC, Springer).

Radhia Cousot was member of the program committees of the 16th International Static Analysis Symposium (SAS'09), the International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC'09), the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM'09). She is on the program committees of the International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC'10), the ACM SIGPLAN 2010 Conference on Programming Language Design and Implementation (PLDI'10), the 38th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'11), and the 20th European Symposium on Programming (ESOP'11). Radhia Cousot is co-PC-chair of the 17th International Static Analysis Symposium (SAS'10) and co-general chair of the 2nd International Workshop on Numerical and Symbolic Abstract Domains (NSAD'10), the 1st Workshop on Static Analysis and Systems Biology SASB'10, and the workshop on Tools for Automatic Program AnalysiS (TAPAS'10).

Jérôme Feret was member of the program committee of the Fifth International Workshop on Developments in Computational Models (DCM 2009) and is co-PC-chair of the 1st Workshop on Static Analysis and Systems Biology (SASB'10).

Laurent Mauborgne is a member of the program committee of the 17th International Static Analysis Symposium (SAS'10).

Antoine Miné is co-PC-chair of the Second International Workshop on Numerical and Symbolic Abstract Domains (NSAD'10).

Xavier Rival was member of the program committee of the eleventh Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2010), the COCV 2009 Workshop (Compiler Optimization meets Compiler Verification), and co-PC chair of the workshop on Tools for Automatic Program AnalysiS (TAPAS'10).

9.1.4. PhD and Habilitation Juries

Bruno Blanchet was reviewer of the PhD thesis of Antoine Mercier (ENS Cachan, December 2009).

Patrick Cousot was in the jury of the PhD thesis of Colas Le Guernic (University of Grenoble, 28 October 2009) and Jean-Baptiste Tristan (University of Paris 7, 6 November 2009).

Radhia Cousot was superviser of the PhD thesis of Pietro Ferrara defended at École polytechnique on 22 May 2009. She was in the jury of the PhD thesis of Anna Zaks (New York University, 29 April 2009).

Jérôme Feret was in the jury of the PhD thesis of Sylvain Pradalier (École Polytechnique and Università di Bologna, September 2009).

9.2. Teaching

9.2.1. Supervision of PhDs and Internships

Patrick Cousot and Antoine Miné supervised the research apprenticeship of Liqian Chen.

Radhia Cousot supervised the PhD thesis of Pietro Ferrara.

Bruno Blanchet supervised the M2 internship [44] of David Cadé (March–July 2009) and is supervising his PhD thesis (September 2009–). Bruno Blanchet is supervising the M2 internship of Miriam Paiola (Università di Padova, September 2009–April 2010) and a research internship of Ben Smyth (PhD student, University of Birmingham, September 2009–February 2010).

Patrick Cousot and Antoine Miné supervised the research apprenticeship of Éric Frichot (first year ENSIMAG student, July 2009).

Patrick Cousot supervised the internship of Jérémy Leconte [61] (MPRI M2, 4 September 2009).

Xavier Rival supervised the M2 internship of Vincent Laviron [60] (March–August 2009).

9.2.2. Graduate Courses

Bruno Blanchet taught 6 hours in the M2 course on Formal methods for concurrency, of Paolo Baldan and Silvia Crafa, Università di Padova, March 2009 [38].

Patrick Cousot and Radhia Cousot were responsible of the M2 course "Abstract interpretation: application to verification and static analysis" at the MPRI (Master Parisien de Recherche en Informatique) [37]. Julien Bertrane, Patrick Cousot, Jérôme Feret, and Antoine Miné participated in the course [37].

Jérôme Feret taught 19 hours in course on "Domain Specific Abstract Interpretation" at the ROPAS (Research on Program Analysis System) group graduate students at Seoul National University [52].

Pietro Ferrara taught 8 hours in the M2 course on Program Analysis and Verification, of Agostino Cortesi, Università di Venezia, March 2009 [57].

9.2.3. Undergraduate Courses

Julien Bertrane gave practical classes of "Programming Languages and Compilation" [36] at the École Normale Supérieure.

Patrick Cousot gave the M1 course "Foundations of abstract interpretation: application to semantics" [49] at the École Normale Supérieure. He gave introductory course to Abstract Interpretation in the Program and Model Analysis (Graduiertenkolleg Programm- Und Modell-Analyse) course common to the Technische Universität München and the Ludwig-Maximilians-Universität München, Munich, Germany [46]; the Summer School on Theory and Practice of Language Implementation University of Oregon, Eugene, Oregon, USA, [45]; the Summer School Marktoberdorf 2009 on Logics and Languages for Reliability and Security, Marktoberdorf, Germany [48]; and the Software verification course, ETH Zürich, Switzerland [47].

Laurent Mauborgne taught algorithmics courses for second year students (L3-M1 level) at École polytechnique, in cooperation with Jean-Marc Staeyert. He also gave a Static analysis [64] course for third year students at École polytechnique (35 hours).

Xavier Rival gave training sessions on "Algorithmics and programming in Java" and on "Principles of Programming Languages" at the École Polytechnique and a lecture on abstract interpretation and static analysis at the École des Mines de Paris.

9.3. PhD theses

Pietro Ferrara defended his PhD on May 22, 2009 on static analysis via abstract interpretation of multithreaded programs [11].

9.4. Participation in Conferences and Seminars

9.4.1. Participation in Conferences

VMCAI: International Conference on Verification, Model Checking and Abstract Interpretation (Savannah, GA, USA, 18-20 January 2009).
Patrick Cousot, Radhia Cousot and Vincent Laviron attended the conference. Patrick Cousot gave an invited talk (in last minute replacement of Alan Emerson) [22] and chaired a session. Vincent Laviron presented [27].

PEPM: ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (Savannah, GA, USA, 19–20 January 2009).
Radhia Cousot attended the conference.

Usable Verification: NSF workshop on "Usable Verification" (Savannah, GA, USA, 20 January 2009).
Patrick Cousot and Radhia Cousot gave an invited talk [31].

POPL: ACM Symposium on Principles of Programming Languages (Savannah, GA, USA, 21–23 January 2009).
Radhia Cousot attended the conference.

COCV: International Workshop “Compiler Optimization meets Compiler Verification” (York, IK, 22 March 2009).
Xavier Rival attended the conference and gave a one-hour invited lecture [66].

CC: International Conference on Compiler Construction (York, UK, 22–29 March 2009).
Xavier Rival attended the conference.

ESOP: European Symposium on Programming (York, UK, 25–27 March 2009).
Xavier Rival attended the conference.

CoSyProofs: Computational and Symbolic Proofs of Security, Spring School and French-Japanese collaboration workshop (Highashi Izu Peninsula, Japan, 6–9 April 2009).
Bruno Blanchet gave an invited talk [39].

MITACS: 2nd Canada-France Workshop on Foundations & Practice of Security (Grenoble, France, 31 May–5 June 2009).
Bruno Blanchet gave an invited talk [40].

APNOC: International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (Paris, France, 22 June 2009).
Radhia Cousot was invited speaker [30].

CAV: 21th International Conference on Computer Aided Verification (Grenoble, France, June 26–July 2, 2009).
Antoine Miné attended.

ASA: 3rd International Workshop on Analysis of Security APIs (Port Jefferson, NY, USA, 10–11 July 2009).
Bruno Blanchet and David Cadé attended the workshop.

ICALP: 36th International Colloquium on Automata, Languages and Programming (Rhodes, Greece, July 5–12, 2009).
Jérôme Feret attended the conference.

CSF: Computer Security Foundations Symposium (Port Jefferson, NY, USA, 8–10 July 2009).
Bruno Blanchet chaired a session. David Cadé attended the conference.

DCM: 5th International Workshop on Developments in Computational Models (Rhodes, July 11, 2009).
Jérôme Feret attended the workshop.

FCC: Workshop on Formal and Computational Cryptography (Port Jefferson, NY, USA, 11–12 July 2009).
Bruno Blanchet chaired a session. David Cadé presented [29].

SAS: 16th International Static Analysis Symposium (Los Angeles, CA, USA, 9–11 August 2009).
Antoine Miné presented [21]. Patrick Cousot and Radhia Cousot attended the conference. Radhia Cousot chaired a session.

Ecrypt-II Summer School On Provable Security (Barcelona, Spain, 7–11 Sept. 2009).
Bruno Blanchet gave an invited talk [41].

GiorgioLevi Workshop “Logic and Energy: A Visionary Inspirator. A Tribute to Giorgio Levi for Forty Years of Research” (23 October 2009). Patrick Cousot and Radhia Cousot gave an invited talk on [15]

SEFM: 7th IEEE International Conference on Software Engineering and Formal Methods (Hanoi, Vietnam, 23–27 November 2009).
Pietro Ferrara presented [24].

APLAS: 7th Asian Symposium on Programming Languages and Systems (Seoul, Korea, 14–16 December 2009).
Vincent Laviron presented [26].

9.4.2. Invitations and Participation in Seminars

Julien Bertrane gave a seminar presenting his post-doc work “Developing temporal abstract domains that prove the temporal specifications of reactive systems” at the university ECNU, Shanghai, China on September the 27th [35].

Bruno Blanchet presented a talk on “CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols” at the Università di Padova, Italy, March 2009 [42] and at the Stony Brook University, NY, USA, July 2009 [43].

Patrick Cousot organizes the seminar “Semantics and Abstract Interpretation” at ENS.

Patrick Cousot visited the Technische Universität München and the Ludwig-Maximilians-Universität München, Munich, Germany, 17–27 May 2009.

Jérôme Feret presented an abstract interpretation framework for analyzing qualitative properties of biological pathways [54] in working group of the Laboratory of Nonlinear Systems (LANOS) at the École Polytechnique de Lausanne. He presented an abstract interpretation framework for reducing differential semantics for biological networks in the seminar of the CEA-list group [51], in the seminar of the Thrust in Reliable Software Research (TRESOR) group at the École Polytechnique de Lausanne [50] and in a interdisciplinary seminar of the department of Dynamics of Membrane Interactions in Normal et Pathological Cells (DIMNP) at the University of Montpellier 2 [53].

Pietro Ferrara presented a talk on static analysis via abstract interpretation of multithreaded programs in seminars at the Swiss Federal Institute of Technology Zurich (ETH Zurich) [59] and at IRISA [58].

Élodie-Jane Sims attended the DS-09301 Dagstuhl seminar on “Typing, Analysis and Verification of Heap-Manipulating Programs” and gave a talk on pointer analysis and separation logic [71].

Antoine Miné gave a talk on the static analysis of run-time errors in parallel embedded C code at the seminar of the “Laboratoire Preuves, Programmes et Systèmes,” Paris 7, France, Novembre 2009 [65].

Xavier Rival attended the ES_PASS workshop (Feb. 2009, Toulouse) and gave a talk on backward analysis [69]. Xavier Rival gave a talk on abstract interpretation and shape analysis at the “Une demi-heure de Science” Seminar at the Comité des Projets of Inria Rocquencourt [68]. Xavier Rival attended the “Fundamentals of Communications and Networks” joint INRIA-Bell Labs Workshop, and gave a lecture on abstract interpretation and shape analysis [67]. Xavier Rival attended the DS-09301 Dagstuhl seminar on “Typing, Analysis and Verification of Heap-Manipulating Programs” and gave a lecture on shape analysis [70].

Laurent Mauborgne was invited to give a talk on disjunctions in static analysis [62] at IMDEA-Software in Madrid in april 2009. Laurent Mauborgne attended the final ES_PASS workshop (Oct. 2009) and gave a talk on the new applications for ASTRÉE[63].

9.5. Short-Term Visitors

Chris Hankin (Imperial College, Mar. 2009), Roberto Giacobazzi (Università di Verona, Jun./Aug. 2009), Daniel Kaestner (Absint, Jan./Fev./Mar. 2009), Stefana Nenova (Absint, Jan./Fev./Mar. 2009), Francesco Ranzato (Università di Padova, Jun. 2009), Mooly Sagiv (Tel-Aviv University, Mar. 2009), Stephan Wilhelm (Absint, Jan./Fev./Mar. 2009).

10. Bibliography

Major publications by the team in recent years

[1] B. BLANCHET. *A Computationally Sound Mechanized Prover for Security Protocols*, in "IEEE Transactions on Dependable and Secure Computing", vol. 5, n^{o} 4, October–December 2008, p. 193–207.

[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), San Diego, California, USA", 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", vol. 277, n^{o} 1–2, 2002, p. 47–103.

[4] P. COUSOT, R. COUSOT. *Temporal Abstract Interpretation*, in "Conference Record of the Twentyseventh Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Boston, Massachusetts, United States", ACM Press, New York, New York, United States, January 2000, p. 12–25.

[5] P. COUSOT, R. COUSOT. *Systematic Design of Program Transformation Frameworks by Abstract Interpretation*, in "Conference Record of the Twentyninth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon, United States", ACM Press, New York, New York, United States, January 2002, p. 178–190.

[6] 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, Los Angeles, California", ACM Press, New York, NY, 1977, p. 238–252.

[7] J. FERET, V. DANOS, J. KRIVINE, R. HARMER, W. FONTANA. *Internal coarse-graining of molecular systems*, in "Proceedings of the National Academy of Sciences", vol. 106, n^{o} 16, April 2009, p. 6453–6458 GB US .

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

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

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] P. FERRARA. *Static analysis via abstract interpretation of multithreaded programs*, École Polytechnique of Paris (France) and University "Ca' Foscari" of Venice (Italy), May 2009, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

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

[13] P. COUSOT, R. COUSOT. *Grammar Semantics, Analysis, and Parsing by Abstract Interpretation*, in "Theoretical Computer Science", 2009, To appear.

[14] P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, X. RIVAL. *Why does* ASTRÉE *scale up?*, in "Formal Methods in Systems Design", November 2009.

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

[16] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. KRIVINE. *Rule-Based Modelling and Model Perturbation.*, in "Transactions on Computational Systems Biology", vol. 11, 2009, p. 116-137 GB US .

[17] J. FERET, V. DANOS, J. KRIVINE, R. HARMER, W. FONTANA. *Internal coarse-graining of molecular systems*, in "Proceedings of the National Academy of Sciences", vol. 106, n^{o} 16, April 2009, p. 6453–6458 GB US .

[18] A. SIMON, A. KING. *The Two-Variable-Per-Inequality Abstract Domain*, in "Higher Order and Symbolic Computation", 2010 GB .

International Peer-Reviewed Conference/Proceedings

[19] M. ABADI, B. BLANCHET, H. COMON-LUNDH. *Models and Proofs of Protocol Security: A Progress Report*, in "21st International Conference on Computer Aided Verification (CAV'09), Grenoble, France", A. BOUAJJANI, O. MALER (editors), Lecture Notes in Computer Science, vol. 5643, Springer, June 2009, p. 35–49 US .

[20] O. BOUISSOU, É. CONQUET, P. COUSOT, R. COUSOT, J. FERET, K. GHORBAL, É. GOUBAULT, D. LESENS, L. MAUBORGNE, A. MINÉ, S. PUTOT, X. RIVAL, M. TURIN. *Space Software Validation using Abstract Interpretation*, in "Proceedings of the International Space System Engineering Conference on Data Systems in Aerospace (DASIA 2009), Istambul, Turkey", vol. SP-669, ESA, May 2009, p. 1–7.

[21] L. CHEN, A. MINÉ, J. WANG, P. COUSOT. *Interval Polyhedra: An Abstract Domain to Infer Interval Linear Relationships*, in "Proceedings of the 16th International Static Analysis Symposium (SAS'09), Los Angeles, CA, USA", LNCS, vol. 5673, Springer, August 2009, p. 309–325 CN .

[22] P. COUSOT, R. COUSOT. *Abstract-Interpretation-based Static Analysis of Safety-Critical Embedded Software (invited talk)*, in "Verification, Model Checking, and Abstract Interpretation, Savannah, GA, USA", N. D. JONES, M. MÜLLER-OLM (editors), 2009.

[23] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. KRIVINE. *Investigation of a biological repair scheme*, in "Proceedings of the ninth Workshop on Membrane Computing, WMC9, Edinburgh, UK, 28–31 July 2008", G. PAUN (editor), LNCS, n^{o} 5391, Springer, Berlin, Germany, 2009, p. 1–12 GB US .

[24] P. FERRARA. *Checkmate: a Generic Static Analyzer of Java Multithreaded Programs*, in "Proceedings of the Seventh IEEE International Conference on Software Engineering and Formal Methods (SEFM 2009)", IEEE Computer Society, November 2009, p. 169–178.

[25] B. JEANNET, A. MINÉ. *Apron: A Library of Numerical Abstract Domains for Static Analysis*, in "Proceedings of the 21th International Conference on Computer Aided Verification (CAV 2009), Grenoble, France", Lecture Notes in Computer Science, vol. 5643, Springer, June 2009, p. 661–667.

[26] V. LAVIRON, F. LOGOZZO. *Refining Abstract Interpretation-Based Static Analyses with Hints*, in "Proceedings of the 7th Asian Symposium on Programming Languages and Systems (APLAS 2009), Seoul, Korea", Z. HU (editor), Lecture Notes in Computer Science, vol. 5904, Springer, December 14-16 2009, p. 343-358 US .

[27] V. LAVIRON, F. LOGOZZO. *SubPolyhedra: A (More) Scalable Approach to Infer Linear Inequalities*, in "Proceedings of the 10th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2009), Savannah, GA, USA", N. D. JONES, M. MÜLLER-OLM (editors), Lecture Notes in Computer Science, vol. 5403, Springer, January 18-20 2009, p. 229-244 US .

Workshops without Proceedings

[28] B. BLANCHET, A. D. JAGGARD, J. RAO, A. SCEDROV, J.-K. TSAY. *Refining Computationally Sound Mechanized Proofs for Kerberos*, in "Workshop on Formal and Computational Cryptography (FCC 2009), Port Jefferson, NY, USA", July 2009 US DE .

[29] D. CADÉ. *From CryptoVerif Specifications to Computationally Secure Implementations of Protocols (Work in Progress)*, in "Workshop on Formal and Computational Cryptography (FCC 2009), Port Jefferson, NY, USA", July 2009.

[30] R. COUSOT. *Abstraction and Approximation in Abstract Interpretation (Invited lecture)*, in "International Workshop on Abstractions for Petri Nets and Other Models of Concurrency, Paris, France", N. SIDOROVA, A. SEREBRENIK (editors), 22 june 2009.

[31] P. COUSOT, R. COUSOT. *Scaling up with abstract interpretation*, in "NSF Workshop on “Usable Verification” (Amir Pnueli organizer), Savannah, GA, USA", 2009.

Scientific Books (or Scientific Book chapters)

[32] B. BLANCHET. *Using Horn Clauses for Analyzing Security Protocols*, in "Formal Models and Techniques for Analyzing Security Protocols", V. CORTIER, S. KREMER (editors), IOS Press, 2010, To appear.

[33] P. COUSOT, R. COUSOT. *A gentle introduction to formal verification of computer systems by abstract interpretation*, NATO Science Series, Series F: Computer and Systems Sciences. IOS Press, 2009.

[34] E. MURPHY, V. DANOS, J. FERET, R. HARMER, J. KRIVINE. *Rule Based Modelling and Model Refinement*, in "Elements of Computational Systems Biology", H. LODHI, S. MUGGLETON (editors), Wiley Book Series on Bioinformatics, 2009 GB .

Other Publications

[35] J. BERTRANE. *Developing temporal abstract domains that prove the temporal specifications of reactive systems*, September the 27th 2009, University ECNU, Shanghai, China.

[36] J. BERTRANE. *Programming Languages and Compilation*, October - December 2009, L3 Practical classes at École Normale Supérieure, Paris.

[37] J. BERTRANE, P. COUSOT, R. COUSOT, J. FERET, A. MINÉ. *Foundations of abstract interpretation: application to semantics*, 2009, M2 course of the MPRI (Master Parisien de Recherche en Informatique).

[38] B. BLANCHET. *Automatic Verification of Cryptographic Protocols in the Formal Model: the Automatic Verifier ProVerif*, March 2009, M2 course, Università di Padova, Italy.

[39] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, April 2009, Computational and Symbolic Proofs of Security, Spring School and French-Japanese collaboration workshop Highashi Izu Peninsula, Japan.

[40] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, June 2009, 2nd Canada-France Workshop on Foundations & Practice of Security, Grenoble, France.

[41] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, September 2009, Summer School On Provable Security, Barcelona, Spain.

[42] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, March 2009, Seminar, Università di Padova, Italy.

[43] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, July 2009, Seminar, Stony Brook University, NY, USA.

[44] D. CADÉ. *Traduction de spécifications en implémentations protocoles*, École Normale Supérieure, Paris, France, August 2009, Masters thesis.

[45] P. COUSOT. *Abstract Interpretation for the programmer, the end-user, and the theoretician*, 23–31 July 2009, Summer School on Theory and Practice of Language Implementation University of Oregon, Eugene, Oregon, USA.

[46] P. COUSOT. *An Informal Introduction to Abstract Interpretation and applications*, 17–27 May 2009, Program and Model Analysis (Graduiertenkolleg Programm- Und Modell-Analyse) course common to the Technische Universität München and the Ludwig-Maximilians-Universität München, Munich, Germany.

[47] P. COUSOT. *An Informal Introduction to Static Analysis and Verification by Abstract Interpretation*, 25 November 2009, Software verification course, ETH Zürich, Switzerland.

[48] P. COUSOT. *Basic concepts of abstract interpretation*, 2–9 August 2009, Summer School Marktoberdorf 2009 "Logics and Languages for Reliability and Security", Marktoberdorf, Germany.

[49] P. COUSOT. *Foundations of abstract interpretation: application to semantics*, 2009, M1 course of the École Normale Supérieure.

[50] J. FERET. *Automatic reduction of ODE semantics for protein-protein interaction networks by abstract interpretation*, 13 February 2009, Seminar: TRESOR — Thrust in Reliable Software Research group, École Fédérale Polytechnique de Lausanne, Switzerland.

[51] J. FERET. *Automatic reduction of ODE semantics for protein-protein interaction networks by abstract interpretation*, 13 January 2009, Seminar: CEA-list, Saclay, France.

[52] J. FERET. *Domain Specific Abstract Interpretation*, 2009, Graduate student course (19h) at Seoul National University.

[53] J. FERET. *Internal coarse-graining of molecular systems*, 9 December 2009, Seminaire interdisciplinaire du DIMNP, Université Montpellier II, France.

[54] J. FERET. *Reachability analysis of rule-based models*, 12 February 2009, Working group: LANOS — Laboratory of NOnlinear Systems, École Fédérale Polytechnique de Lausanne, Switzerland.

[55] J. FERET, H. KOEPPL, T. PETROV. *Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models*, November 2009, Poster presented at the All SystemsX Day, Bern, Switzerland CH .

[56] J. FERET, H. KOEPPL, T. PETROV. *Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models*, December 2009, Poster presented at: Paris Interdisciplinary PhD Symposium: Frontiers in Life Sciences Graduate School CH .

[57] P. FERRARA. *Numerical Domains*, March 2009, M2 course, Università di Venezia, Italy.

[58] P. FERRARA. *Static analysis by abstract interpretation of Java multithreaded programs*, 25 August 2009, IRISA-INRIA, Rennes, France.

[59] P. FERRARA. *Static analysis by abstract interpretation of Java multithreaded programs*, 29 January 2009, Chair of Programming Methodology, ETH, Zurich, Switzerland.

[60] V. LAVIRON. *Application d'une analyse de formes à un modèle mémoire réaliste*, École Normale Supérieure, Paris, France, August 2009, Masters thesis.

[61] J. LECONTE. *Hiérarchie de sémantique par interprétation abstraite et preuves formelles*, École Normale Supérieure, Paris, France, August 2009, Masters thesis.

[62] L. MAUBORGNE. *Disjunctions that Scale Up*, March 2009, IMDEA Workshop, Madrid, Spain.

[63] L. MAUBORGNE. *New Domains for* ASTRÉE, October 2009, ES_PASS Workshop, Madrid, Spain.

[64] L. MAUBORGNE. *Static Analysis of Programs*, 2009, M1 course of the École Polytechnique.

[65] A. MINÉ. *Static analysis of run-time errors in parallel embedded C code*, November 2009, Seminar, Laboratoire Preuves, Programmes et Systèmes, Paris 7, France.

[66] X. RIVAL. *A Framework for Certified Compilation*, March 2009, COCV Workshop, York, UK.

[67] X. RIVAL. *Abstract interpretation-based static analysis of programs*, June 2009, Bell Labs, Murray Hill, USA.

[68] X. RIVAL. *Analyse statique par interprétation abstraite*, April 2009, INRIA, Rocquencourt, France.

[69] X. RIVAL. *Extension of* ASTRÉE *with backward analysis*, February 2009, ES_PASS Workshop, Toulouse, France.

[70] X. RIVAL. *Shape Analysis Applied to C Code*, July 2009, Shloss Dagstuhl, Germany.

[71] É.-J. SIMS. *A glimpse of my Ph.D.: pointer analysis and separation logic*, July 2009, Shloss Dagstuhl, Germany.

References in notes

[72] B. BLANCHET. *Computationally Sound Mechanized Proofs of Correspondence Assertions*, in "20th IEEE Computer Security Foundations Symposium (CSF'07), Venice, Italy", IEEE, July 2007, p. 97–111.

[73] B. BLANCHET. *A Computationally Sound Mechanized Prover for Security Protocols*, in "IEEE Transactions on Dependable and Secure Computing", vol. 5, n^{o} 4, October–December 2008, p. 193–207.

[74] B. BLANCHET, A. D. JAGGARD, A. SCEDROV, J.-K. TSAY. *Computationally Sound Mechanized Proofs for Basic and Public-key Kerberos*, in "ACM Symposium on Information, Computer and Communications Security (ASIACCS'08), Tokyo, Japan", ACM, March 2008, p. 87–99 US .

[75] L. CHEN, A. MINÉ, P. COUSOT. *A Sound Floating-Point Polyhedra Abstract Domain*, in "Proc. of the Sixth Asian Symposium on Programming Languages and Systems (APLAS'08), Bangalore, India", Lecture Notes in Computer Science, vol. 5356, Springer, December 2008, p. 3–18.

[76] 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, NY, USA, 2007, p. 7–9.

[77] 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, Thèse d'État ès sciences mathématiques.

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

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

[80] 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, vol. 4444, Springer, Berlin, Germany, 2007.

[81] 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, New York, United States, 1979, p. 269–282.

[82] 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, vol. 3444, Springer, Berlin, Germany, 2–10 April 2005, p. 21–30.

[83] 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,

Shanghai, China", M. HINCHEY, J. HE, J. SANDERS (editors), IEEE Computer Society Press, Los Alamitos, California, USA, 6–8 June 2007.

[84] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. KRIVINE. *Rule-based modelling, symmetries, refinements.*, in "Proceedings of the First International Workshop, Formal Methods in Systems Biology, FMSB'2008, Cambridge, UK", J. FISHER (editor), Lecture Notes in BioInformatics, vol. 5054, Springer, Berlin, Germany, 4–5 June 2008, p. 103–122 GB US .

[85] A. SIMON, A. KING, J. M. HOWE. *Two Variables per Linear Inequality as an Abstract Domain*, in "Logic-Based Program Synthesis and Transformation, Madrid, Spain", M. LEUSCHEL (editor), LNCS, vol. 2664, Springer, September 2003, p. 71–89 UK .