INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Abstraction

Abstract Interpretation

Paris - Rocquencourt

THEME SYM

Activity Report

2007

Table of contents

1. Team

Head of project-team
Patrick Cousot [Professor/Professeur, ENS, HdR]

Research scientists
Bruno Blanchet [CR, CNRS]
Radhia Cousot [DR, CNRS, HdR]
Laurent Mauborgne [Assistant Professor/Maître de conférences, ENS, HdR]
Antoine Miné [CR, CNRS]
David Monniaux [CR, CNRS, Jan. — Aug. 2007]
Xavier Rival [CR, INRIA Paris–Rocquencourt]

PhD students
Julien Bertrane [ENS]

Post-doctoral fellows
Jérôme Feret [ENS]
Axel Simon [University of Kent, 1 Nov. 2007 —]

Administrative assistant
Nathalie Gaudechoux [INRIA]
Joëlle Isnard [Administrative Head DI, ENS]

Student interns
Ferdinanda Camporesi [Università di Bologna, 1 Oct. 2007 —]
Liquian Chen [1 Oct. 2007 —]
David Durrleman [ENS, 15 Sep. 2007 —]

Technical staff
Élodie-Jane Sims [Research engineer, ENS, 1 Oct. 2007 —]

Visiting scientist
Roberto Giacobazzi [Università di Verona]
Benjamin Goldberg [New York University]

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, develop, analyze, and verify highly complex systems, from computer-based to biological ones. They also underlie the design of safety and security partial verification *tools*.

2.2. Highlights of the Year

In 2007, the team became an INRIA ABSTRACTION project-team.

In 2007, the decision has been taken to consider the industrialization of the ASTRÉE static analyzer for the verification of absence of runtime errors in embedded synchronous control/command code. The inclusion of ASTRÉE in the development of the critical software for the future A350 would be the first industrial application following the successful applications for the A340 and A380 in a research and development context.

3. Scientific Foundations

3.1. Abstract Interpretation Theory

The abstract interpretation theory [54], [57], [58] 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 relative 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.

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 are undecidable, and so, automatic tools for reasoning about programs are all incomplete (for non-trivial program properties such as safety, liveness, or security) and must therefore fail on some programs. This can be achieved by allowing the tool not to terminate, 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. [24], [25]) or incomplete (e.g. [2]).

Sound, terminating and precise tools are difficult to design. Complete tools to solve non-trivial verification problems are impossible to design, 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 [59]. 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 [57], [58];

- guiding the correct formal design of automatic tools for *program analysis* (computing an abstract semantics) and *program verification* (proving that an abstract semantics satisfies an abstract specification) [55].

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 information about the runtime behavior of programs. For example the ASTRÉE static analyzer 5.1, which was developed by the team these last six years, aims at proving the absence of runtime errors in programs written in the C programming language. It is used in the avionics 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" [56] 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; 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, sound program analysis methods such as abstract-interpretation-based static analysis are able to cope with these programs, since they can prove the absence of bugs. Yet, these techniques are generally incomplete since the absence of runtime errors is undecidable in practice; therefore, they are prone to false alarms (*i.e.*, they may fail to prove the absence of runtime errors for a program which is safe).

It should be noted that, due to the size of the critical codes (typically above 100 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 (hence, producing a full proof of absence of runtime error) 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 [21], [26], which has been initially designed specifically for synchronous embedded software.

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

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 for six years, using abstract interpretation techniques. Successful results have been achieved on industrial applications. The ABSTRACTION project-team has strong plans to continue research on this topic and to industrialize ASTRÉE.

4.2. Security Protocols

Security protocols use cryptography in order to guarantee the security of exchanges over an insecure network, such as 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 lead 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 Signalling Networks

Cell Signalling Networks models suffer from a combinatorial blow up in the number of species (number of non-isomorphic ways in which some proteins can connect to each others). This large number of species makes the design and the analysis of these models a highly difficult task.

Contextual graph-rewriting systems allow a concise description of these networks, which leads to a scalable method for modelling them. Then abstract interpretation allows the abstraction of the properties of these systems. It provides debugging information in the design phases. It also provides static information to abstract their global properties. Then, these properties are necessary in order to make other computations scale up. For instance, ODE (Ordinary Differential Equations) generation, stochastic simulations and calibration may be considered without loss of information after this appropriate abstraction.

5. Software

5.1. The Astrée Static Analyzer

Keywords: *absence of runtime error*, *abstract interpretation*, *static analysis*, *verifier*.

Participants: Patrick Cousot [project leader, correspondant], Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival, Bruno Blanchet [Nov. 2001–Nov. 2003], David Monniaux [Nov. 2001–Aug. 2007].

The ASTRÉE static analyzer [21], [26] 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 lead to consider the inclusion of ASTRÉE in the production of the critical software for the A350.

5.2. The Apron Numerical Abstract Domain Library

Keywords: *convex polyhedron*, *interval*, *linear equality numerical abstract domain*, *octagon*.
Participants: Antoine Miné [correspondant], 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, and provide a teaching and demonstration tool to disseminate knowledge on abstract interpretation.

The APRON library is not tied to a particular numerical abstraction. Several abstract domains providing various precision versus cost trade-offs are currently implemented: the interval, the octagon, and the polyhedron domain. A specific low-level C API was designed to minimize the effort when incorporating a new abstract domain: only the basic functions need to be implemented. The library contains generic services and fallback functions to simplify domain integration. For instance, existing domains can be combined by instancing a generic reduced product construction with minimal effort. Another example is the support for non-linear transfer functions for free through a generic linearization technique. The API exposes domain-independent data-types and supports both multi-precision and machine floating-point numbers. It is thread-safe.

From the point of view of the analysis designer, the APRON library exposes a higher-level, richer, and language-agnostic API. Bindings for C, C++, and OCaml are currently provided.

In order to disseminate the knowledge in abstract interpretation, a simple interprocedural static analyzer for a toy language has been designed and deployed through a web-interface: http://pop-art.inrialpes.fr/interproc/interprocweb.cgi.

The APRON library is freely available on the web at http://apron.cri.ensmp.fr/library. It is released under the LGPL license. Work on the APRON library started during the APRON project 6.11 and is progressing rapidly. The core library (not counting language bindings) is now 24000 lines of C (more than doubling since last year) and this year has seen releases 0.9.5 to 0.9.8.

Current external users includes 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).

5.3. Translation Validation

Keywords: *abstract interpretation*, *certified compilation*, *static analysis*, *translation validation*, *verifier*.

Participant: Xavier Rival [correspondant].

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, which ensures 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

Keywords: *formal model*, *security protocols*, *verifier*.

Participants: Bruno Blanchet [correspondant], Xavier Allamigeon [April–July 2004].

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 that 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.

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

5.5. CryptoVerif

Keywords: *computational model*, *security protocols*, *verifier*.

Participant: Bruno Blanchet [correspondant].

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;
- correspondences [17], which include in particular authentication; this is the main extension implemented this year.

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 and a project for using it as a back-end for verifying implementations of protocols in F# is starting at Microsoft Research Cambridge.

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

6. New Results

6.1. Abstract Semantics of Grammars

Keywords: *abstract semantics*, *bottom-up semantics*, *context-free grammar*, *grammar flow analysis*, *grammar problem*, *parsing*, *top-down semantics*.

Participants: Patrick Cousot, Radhia Cousot.

We have introduced abstract interpretations of a fixpoint protoderivation semantics defining the maximal derivations of a transitional semantics of context-free grammars akin to pushdown automata. The result is a hierarchy of bottom-up or top-down semantics refining the classical equational and derivational language semantics and including Knuth grammar problems, classical grammar flow analysis algorithms, and parsing algorithms [25].

6.2. 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 [24].

6.3. 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 mainly worked on the proof of correspondence properties in PROVERIF and on case studies that use this verifier.

6.3.1. Automatic Verification of Correspondences for Security Protocols

Keywords: *authentication*, *automatic verification*, *correspondences*, *formal model*, *security protocols*.

Participant: Bruno Blanchet.

We have written a journal paper that summarizes the technique used by PROVERIF in order to verify correspondences in security protocols. Correspondences are properties of the form “if some event has been executed, then other events have been executed as well”. In particular, correspondences can be used to formalize authentication. This technique can prove a wide variety of correspondence properties represented by particular logical formulae. It is fully automatic, it handles an unbounded number of sessions of the protocol, and it is efficient in practice. It significantly extends a previous technique for the verification of secrecy. The protocol is represented in an extension of the pi calculus with fairly arbitrary cryptographic primitives. This protocol representation includes the specification of the correspondence to be verified, but no other annotation. This representation is then translated into an abstract representation by Horn clauses, which is used to prove the desired correspondence. This technique has been proved correct and tested on various protocols from the literature. The experimental results show that these protocols can be verified by our technique in less than 1 s. It has also been used in more ambitious case studies such as our case studies of JFK [15] and Plutus.

6.3.2. Case Study: the Protocol Just Fast Keying (JFK)

Keywords: *Just Fast Keying*, *applied pi calculus*, *automatic verification*, *security protocols*.

Participants: Martín Abadi [University of California, Santa Cruz and Microsoft Research Silicon Valley], Bruno Blanchet, Cédric Fournet [Microsoft Research Cambridge].

JFK is a recent, attractive protocol for fast key establishment as part of securing IP communication. We have analyzed it formally in the applied pi calculus (partly in terms of observational equivalences, partly with the assistance of the automatic protocol verifier PROVERIF). We have treated JFK's core security properties, and also other properties that are rarely articulated and studied rigorously, such as plausible deniability and resistance to denial-of-service attacks. In the course of our analysis we found some ambiguities and minor problems, such as limitations in identity protection, but we mostly obtained positive results about JFK. For this purpose, we developed ideas and techniques that should be useful more generally in the specification and verification of security protocols. This work is published in TISSEC [15].

6.3.3. Case Study: the Secure Storage System Plutus

Keywords: *automatic verification*, *lazy revocation*, *secure storage*, *security protocols*.

Participants: Bruno Blanchet, Avik Chaudhuri [University of California, Santa Cruz].

We have studied formal security properties of the state-of-the-art protocol for secure file sharing on untrusted storage Plutus, in the automatic protocol verifier PROVERIF. As far as we know, this is the first automated formal analysis of a secure storage protocol. The protocol used as the basis of Plutus features a number of interesting schemes like lazy revocation and key rotation. These schemes improve the protocol's performance, but complicate its security properties. Our analysis clarifies several ambiguities in the design and reveals some unknown attacks on the protocol. We propose corrections, and prove precise security guarantees for the corrected protocol.

6.4. 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 extended it to correspondence assertions and used it in case study of Kerberos.

6.4.1. Computationally Sound Mechanized Proofs of Correspondence Assertions

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

Participant: Bruno Blanchet.

We have extended our mechanized prover CRYPTOVERIF for showing correspondence assertions for security protocols in the computational model. Correspondence assertions are useful in particular for establishing authentication. Our technique produces proofs by sequences of games, as standard in cryptography. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary. Our technique can handle a wide variety of cryptographic primitives, including shared- and public-key encryption, signatures, message authentication codes, and hash functions. It has been successfully tested on examples from the literature, and used in the case study of Kerberos mentioned below. This work has been presented at Computer Security Foundations Symposium [17].

6.4.2. 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], Andre Scedrov [University of Pennsylvania], Joe-Kai Tsay [University of Pennsylvania].

We have done a computationally sound mechanized analysis of Kerberos 5, both with and without its public-key extension PKINIT. We have proved authentication and key secrecy properties using the prover CRYPTOVERIF, which works directly in the computational model; these are the first mechanical proofs of a full industrial protocol at the computational level. We also generalize the notion of key usability and use CRYPTOVERIF to prove that this definition is satisfied by keys in Kerberos. This work has been presented at the Dagstuhl seminar “Formal Protocol Verification Applied” [33] and is to appear at AsiaCCS'08.

6.5. Analysis of Biological Pathways

We have introduced a framework to design and analyze biological networks. We focus on protein interaction networks described as graph rewriting systems. Such networks can be used to model some signalling 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).

6.5.1. Reachability Analysis of Biological Signalling Pathways by Abstract Interpretation

Keywords: *protein interaction networks. verification.*

Participant: Jérôme Feret.

We have developed an abstract interpretation-based framework to compute an over-approximation of the reachable species in protein interaction networks. We show several applications in [30]: first, we use this abstraction to detect some bugs such as dead reactions (reactions that can never be triggered) and conflicting rules (distinct rules that compute the same thing); our analysis also predicts whether two sites may bind in any context, or if this binding is controlled by other sites. This analysis can be used to debug models, and to check that they match with what the biologist has in mind.

6.5.2. Scalable Simulation of Cellular Signaling Networks

Keywords: *protein interaction networks*, *stochastic simulation*.

Participants: Vincent Danos [Paris VII], Jérôme Feret, Walter Fontana [Harvard Medical School], Jean Krivine [École Polytechnique].

We have introduced a Gillespi simulation algorithm for the protein interaction networks that are described as graph rewriting systems. This algorithm does not count the number of species, but uses a radically different method. The proposed algorithm uses a representation of each protein of the system together with an over approximation of the potential embedding of the rewriting rules in this system, and a specific correction scheme to obtain exact timing. The update of the potential embedding of the rewriting rules is computed efficiently thanks to our reachability analysis. Being completely local, this algorithm has a per event time cost which is independent of the size of reachable species (which can even be infinite), and independent of the size of the system. The per event time cost is only logarithmic with respect to the number of rewriting rules. Nevertheless, the Gillespi time (*i.e.* the biological time progress at each simulation step) is inversely proportional to the size of the system. We have published this algorithm in [29].

6.5.3. Rule-based Modelling of Cellular Signalling

Keywords: *concurrency*, *epidermic growth factor model*, *protein interaction networks*.

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

We have used our framework to model a sizable protein interaction network obtained from refactoring two models of EGF (epidermic growth factor) receptor signalling that are based on differential equations. In [28], we have published this model and we have showed that an exciting aspect of our modelling approach is that it naturally lends it-self to the identification and analysis of the causal structures that deeply shape the dynamical, and perhaps even evolutionary, characteristics of complex distributed biological systems. In particular, we have adapted the notions of causality and conflict, familiar from concurrency theory, to graph rewriting systems. Using the EGF receptor model as an example, we have showed how causality enables the formalization of the colloquial concept of pathway and, perhaps more surprisingly, how conflict can be used to dissect the signalling dynamics to obtain a qualitative handle on the range of system behaviours. We have showed that by taming the combinatorial explosion, and exposing the causal structures and key kinetic junctures in a model, agent- and rule-based representations hold promise for making modelling more powerful, more perspicuous, and of appeal to a wider audience.

6.6. Representation of Sets of Graphs

Keywords: *graphs*, *sharing representations*, *static analysis*, *symbolic abstract domains*.

Participant: Laurent Mauborgne.

In order to derive generic abstract domains for symbolic properties where no hierarchy property can be extracted, we developed efficient representations for sets of graphs [13]. These representations extend our previous work on sets of trees, generalizing old techniques and introducing new implementation issues. They allow for a sharing representation which was shown to be experimentally superior to the finite height hashing technique. They make new use of partitioning algorithms *a la* Hopcroft to compute an ordering on graph nodes in $O(n \log n)$. New widenings have been devised. Combined with the efficient representation, they should lead to faster and more precise analyses on symbolic, non hierarchic structures, such as heap shapes or network shapes.

6.7. The Trace Partitioning Abstract Domain

Keywords: *absence of runtime error*, *disjunctions*, *static analysis*, *symbolic abstract domains*, *trace partitioning*.

Participants: Laurent Mauborgne, Xavier Rival.

In order to achieve better precision of abstract interpretation based static analysis, we introduced a new generic abstract domain, the trace partitioning abstract domain. We extended this principle into a theoretical framework [16] allowing a wide range of instantiations of the domain. We proved that all these instantiations give correct results. We studied the properties of this family of abstractions. Last, we implemented and tuned an instance of this generic abstract domain, and integrated it into the ASTRÉE static analyzer. The domain can be configured so as to cope with program specificities. It allows for an important gain in both performance and precision.

6.8. Shape Analysis with Structural Invariant Checkers

Keywords: *absence of runtime error*, *inductive definitions*, *memory abstraction*, *shape analysis*, *static analysis*, *symbolic abstract domains*.

Participants: Bor-Yuh Evan Chang [University of California at Berkeley (USA)], George Necula [University of California at Berkeley (USA)], Xavier Rival.

Developer-supplied data structure specifications are important to shape analyses, as they tell the analysis what information should be tracked in order to obtain the desired shape invariants. We observe that data structure checking code (*e.g.*, used in testing or dynamic analysis) provides shape information that can also be used in static analysis. We proposed a lightweight, automatic shape analysis based on these developer-supplied structural invariant checkers [18]. In particular, we set up a parametric abstract domain, which is instantiated with such checker specifications to summarize memory regions using both notions of complete and partial checker evaluations. The analysis then automatically derives a strategy for canonicalizing or weakening shape invariants.

6.9. Separation Analysis

Keywords: *heap*, *modular analysis*, *pointer analysis*, *separation analysis*, *static analysis*.

Participant: Élodie-Jane Sims.

The objective is to perform a modular static analysis of programs manipulating data structures involving dynamic data allocation and pointers. *Separation logics* is a recently developed logic to describe and express properties of the memory. The logic was first extended with fixpoints to express recursive properties so as to provide more precise pre- and post-conditions rules for while loops as well as strongest postconditions. The next goal was to use separation logic with fixpoints as an interface language for pointer analyses (for example shape analyses). As a first step, we designed an abstract domain in the form of an abstract language with similarities with shape graphs as well as several useful abstract operations. It was given a concrete semantics in terms of sets of memory, so as to express and prove overapproximating translations of formulae of the extended separation logic into this abstract language [14].

6.10. Analysis of a USB Driver

Keywords: *USB*, *absence of runtime error*, *asynchronous*, *concurrency*, *device driver*, *intelligent device static analysis*, *verifier*.

Participants: David Monniaux [until Aug. 2007], Antoine Miné [Sept.–Dec. 2007].

We have studied the automated proof of absence of runtime errors for a USB (Universal Serial Bus) driver included in an embedded safety-critical application. The driver communicates with several USB controllers that are 'intelligent' pieces of hardware using linked lists of descriptors for pending memory transfers. The controllers run concurrently with the driver. They inspect and modify the descriptor lists residing in shared memory (relying solely on the atomicity of word reads and writes for synchronization) and implement data transfers using direct memory access to and from the address space of the main software. To prove the memory safety of the driver, it is thus critical to take the intelligent controllers into account and prove that the driver does not misprogram them (*e.g.*, ordering them to overwrite some important memory locations). Because of the critical level of the software, the considered USB driver was much simpler than drivers that can be found in consumer-level operating systems: it lacked complex USB features such as hot-plugging and refrained from using dynamically memory allocation, which made the analysis simpler.

To perform a static analysis of the provided USB driver, we first designed a model of a controller respecting the OHCI (Open Host Controller Interface) norm 1.0a. This model is a 450-line program written in a subset of C, enriched with non-deterministic choice (used pervasively to abstract away from unspecified implementation details as well as fully specified behaviors not relevant to the proof of memory safely, such as the order of certain operations, time-related properties, etc.). We then extended the ASTRÉE analyzer (5.1) to automatically insert computation steps for the controllers in-between relevant execution steps of the driver (more precisely, each time the driver observes or modify the shared memory state), thus achieving the asynchronous combination of the driver and the controllers. Finally, ASTRÉE was enriched to permit the precise analysis of pointers, including bitwise operations for address masking and synthesis, in a context where all memory is statically allocated.

Unfortunately, some of the properties required to prove the absence of runtime errors involved complex separation properties of linked lists, which are far outside the current scope of ASTRÉE. Inspection by hand concluded that the model of the controllers could be simplified by removing behaviors that are expected by the OHCI norm but do not change the set of reachable states for live variables. ASTRÉE was then able to prove automatically the absence of arithmetic and memory errors for the driver and the simplified controllers.

This work was made under the ASBAPROD contract (7.3) and has been published in EMSOFT [31].

6.11. Numerical Abstract Domains in the Apron Library

Keywords: *congruence equality*, *convex polyhedron*, *floating-point arithmetic*, *linear equality*, *linearization*, *numerical abstract domain*, *octagon*, *reduced product*, *static analysis*.

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

Several features were added this year to the APRON numerical abstract domain library (5.2).

A core new feature is the introduction of transfer functions for *non-linear* assignments and tests. A new expression tree data-type was added to support all four operations as well as square root, modulo and casts; with a choice of integer, real or IEEE 754-1984 *floating-point* semantics (with optional rounding mode). The main result concerns the ability for all numerical abstract domains to support non-linear transfer functions for free through the use of generic, domain-independent linearization techniques. An important application is the ability for complex relational numerical abstract domains on reals (such as polyhedra or octagons) to support floating-point expressions in a sound way. We hope that this APRON-specific feature will simplify the design of sound static analyzers for real-life programming languages.

Moreover, the following numerical abstract domains were added the library:

- a linear equality domain, based on the NewPolka polyhedra;
- wrappers for the polyhedra and linear congruences domains from the Parma Polyhedra Library (PPL);
- a generic reduced product domain constructor;
- an instance of the above constructor for the reduced product of convex polyhedra (NewPolka implementation) and linear congruences (PPL implementation).

One new language binding was added: C++ that adds object-orientation, intelligent constructors and destructors, and operator overloading to provide a more user-friendly API.

An interprocedural analyzer for a toy language has been developed in OCaml and made available on the web at http://pop-art.inrialpes.fr/interproc/interprocweb.cgi. It provides a non-trivial example of the use of the APRON library and an academic tool to disseminate the knowledge on numerical abstract domains and abstract interpretation.

Dissemination of the library was also achieved this year through a poster at the Static Analysis Symposium (SAS'07) [50] as well as a talk and tool presentation at the "Grand Colloque STIC 2007".

This year has also seen the emergence of an international community of users from various research labs (see 5.2 for a list). We provided support for these users and corrected the reported bugs.

6.12. Abstraction Refinement

Keywords: *abstraction*, *fixpoint*, *reachability*, *refinement*.

Participants: Patrick Cousot, Pierre Ganty [Université Libre de Bruxelles], Jean-François Raskin [Université Libre de Bruxelles].

New fixpoint guided abstraction refinement algorithms for abstract reachability in finite transition systems were designed in cooperation with the ULB (Université Libre de Bruxelles) [27]. The abstract fixpoint checking algorithm performs an automatic refinement by backward completion in Moore closed abstract domains. The algorithm is more precise than the counter-example guided abstract refinement algorithm (CEGAR). Contrary to several works in the literature, the algorithm does not require the abstract domains to be partitions of the state space. The proposed automatic refinement technique is compatible with so-called acceleration techniques. Furthermore, the use of Boolean closed domains does not improve the precision of the algorithm. The algorithm has been illustrated by proving properties of programs with nested loops.

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.

7.2. SSVAI Contract

SSVAI (Space Software Validation using Abstract Interpretation) is an ESA-ITI project (European Space Agency (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).

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 results in 2007 concerned a new parallel version of ASTRÉE and the static analysis of USB (universal serial bus) drivers in the context of a simulated concurrent device controler [31].

7.4. Apron ACI

The ABSTRACTION project-team participates in a French *Action Concertée Incitative "Sécurité et Informatique"* (ACI SI) named "Apron" http://apron.cri.ensmp.fr/ together with the team "Analyses, transformations et instrumentations de programmes" (Centre de Recherche en Informatique, École des Mines de Paris [coordinator]), the SYNCHRONE team (Verimag, Grenoble), the VERTECS project (IRISA, Rennes), and the team "Sémantique, preuves et interprétation abstraite" (École Polytechnique, Palaiseau). The focus of this project is on the theory of numerical abstract domains and their application to the static analysis by abstract interpretation of the properties of the numerical variables of a program. The first, theoretical goal of the project is to advance the research in numerical abstract domains. The second, more practical goal, is to mature the field by bringing together five actors to define their needs, and then design and implement a common software platform suited for a broad range of static analysis applications. This project has lead to 29 publications in international conferences, workshops, and journals, four technical reports, four PhD, and the design and implementation of the APRON numerical abstract domain library (5.2). The project started in October 2004 and ended in October 2007.

7.5. 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 opened 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.

7.6. 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 (January 2006–December 2008) brings together researchers of 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 lead to 15 publications in international conferences, workshops, and journals, to the implementation of two tools, CRYPTOVERIF and an extension of AVISPA, and strongly contributed to the organisation of a series of international workshops (the workshops on Formal and Computational Cryptography, FCC). Bruno Blanchet is the principal investigator for this action.

7.7. 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.

8. Dissemination

8.1. Interaction with the Scientific Community

8.1.1. Collective Responsibilities

Bruno Blanchet is a member of the *commission de spécialistes* (hiring committee) of ENS Cachan.

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

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

8.1.2. Editorial Boards and Program Committees

Bruno Blanchet is associate editor of the International Journal of Applied Cryptography (IJACT). He was member of the program committee of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2007), the IEEE Computer Security Foundations Symposium (CSF 2007), the 18th International Conference on Concurrency Theory (CONCUR 2007), and the 3rd Workshop on Formal and Computational Cryptography (FCC 2007).

Patrick Cousot is a member of the Academia Europaea. He was the 2007 chair of the ACM Grace Murray Hopper Award Committee. He is member of the IFIP working group WG 2.3 on programming methodology. He 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. He is a member of the advisory board of the Higher-Order Symbolic Computation journal (HOSC, Springer) and of the Asian Association for Foundations of Software (AAFS), member of the steering committees of the Static Analysis Symposium (SAS) and Verification, Model-Checking and Abstract Interpretation (VMCAI) conferences, member of the program committees of the Static Analysis Symposium (SAS'07), European Symposium on Programming (ESOP'07), and Principles of Programming Languages (POPL'07) conferences.

Radhia Cousot is member of the advisory board of the Higher-Order Symbolic Computation journal (HOSC, Springer) and member of the program committee of the Verification, Model Checking and Abstract Interpretation (VMCAI'07) conference. Radhia Cousot is head of the Abstract Interpretation group at École Polytechnique under a convention between the CNRS, the École Normale Supérieure and the École Polytechnique.

Antoine Miné participated in the program committee of the 16th International Conference on Compiler Construction (CC 2008) and the 10th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2007).

8.1.3. PhD and Habilitation Juries

Bruno Blanchet was a member of the PhD jury of Mathieu Baudet (ENS Cachan, January 2007).

Patrick Cousot was the director of the HdR of Laurent Mauborgne (University Paris-Dauphine, February 2007), and reviewer and member of the PhD jury of Mila Dallapreda (Università di Verona, May 2007) and of Pierre Ganty (Universié Libre de Bruxelles, September 2007).

Radhia Cousot was director of the PhD thesis of Élodie-Jane Sims (École polytechnique, December 2007)

Laurent Mauborgne was the external examiner for the PhD thesis of Neil Kettle (University of Kent, September 2007).

8.2. Teaching

8.2.1. Supervision of PhDs and Internships

Patrick Cousot supervised the PhD thesis of Julien Bertrane and the research apprenticeships of Ferdinanda Camporesi, Liquian Chen, and David Durrleman.

8.2.2. Training

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

The ABSTRACTION project-team organised a one-day training session on the ASTRÉE static analyzer (5.1) for academic and industrial partners in the ES_PASS project (7.1).

8.2.3. Research Courses

Patrick Cousot gave a course on abstract interpretation at the IBM Thomas J. Watson Research Center [36].

Patrick Cousot, Laurent Mauborgne, Antoine Miné, and Xavier Rival gave a one day training session for end-users of ASTRÉE at the École Normale Supérieure [46].

8.2.4. Graduate Courses

Bruno Blanchet was co-responsible with Steve Kremer [INRIA, ENS Cachan] of the MPRI (Master Parisien de Recherche en Informatique) course on Cryptographic protocols: formal and computational proofs, and he taught 12 hours in this course.

Radhia Cousot was responsible of the M2 course “Abstract interpretation: application to verification and static analysis” at the MPRI (Master Parisien de Recherche en Informatique) [44]. Patrick Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, and Xavier Rival participated in the course.

Jérôme Feret gave a 1-hour and half lecture and a 1-hour and half practical session on formal biology at the MPRI (Master Parisien de Recherche en Informatique).

Antoine Miné gave a 2-hour lecture on the ASTRÉE static analyzer at the Master Ingénierie des Systèmes Industriels Complexes (École Polytechnique, Palaiseau).

Xavier Rival gave a 2-hour lecture on abstract interpretation based static analysis at the University of California at Berkeley.

Laurent Mauborgne was a partial time assistant professor (*professeur chargé de cours*) at École Polytechnique. He gave a 32-hour course on static analysis for 3rd year students (M1). He gave 30 hours of lectures in small groups for 2nd year students, following the course “Foundations of Computer Science” directed by François Morain and Jean-Marc Steyaert.

8.2.5. Undergraduate Courses

Julien Bertrane was teaching assistant at École Polytechnique.

Patrick Cousot gave the L3 course “Compilation and Programming Languages” [45] and the M1 course “Foundations of abstract interpretation: application to semantics” [37] at the École Normale Supérieure.

Laurent Mauborgne gave training sessions for the course “Compilation and Programming Languages” at the École Normale Supérieure [45].

Antoine Miné was a temporary assistant professor (*attacheé temporaire d’enseignement et de recherche*) at the École Normale Supérieure until September 2007. He gave training sessions for the “Algorithms and programming” and the “System and networks” courses. He organised and gave a 25-hour course on C programming for non computer scientists.

Xavier Rival gave training sessions on “Algorithmics and programming in Java” at the École Polytechnique and a 4-hour lecture on abstract interpretation and static analysis at the École des Mines de Paris.

8.3. Participation in Conferences and Seminars

8.3.1. Participation in Conferences

ACM Awards (San Diego, Juin 2007). Patrick Cousot chaired the ACM Grace Hopper Award Committee in 2007 and participated in the award ceremony.

ASTReNet: Formal Aspects of Source Code Analysis and Manipulation (London, UK, March 2007).
Patrick Cousot gave an invited talk at the Thirteenth ASTReNet Workshop “Formal Aspects of Source Code Analysis and Manipulation” [40].

CSF: Computer Security Foundations Symposium (Venice, Italy, July 2007).
Bruno Blanchet presented [17] and chaired a session.

EMSOFT (Salzburg, Austria, Sep. 2007).
Patrick Cousot and Radhia Cousot. Patrick Cousot gave an invited tutorial at the Seventh ACM & IEEE International Conference on Embedded Software [21]. David Monniaux presented [31].

ES_PASS Workshop (Berlin, Germany, Oct. 2007].
Patrick Cousot, Radhia Cousot, Antoine Miné, Élodie-Jane Sims. Patrick Cousot gave a presentation of abstract interpretation and the ASTRÉE static analyzer.

FCC: Workshop on Formal and Computational Cryptography (Venice, Italy, July 2007).
Bruno Blanchet chaired a session.

FOSSACS: Foundations of Software Science and Computation Structures (Braga, Portugal, March 2007).
Antoine Miné chaired a session.

IBM Programming Language Day (Hawthorn, NY, USA, May 2007).
Patrick Cousot gave a talk [42].

ISOLA (Futuroscope, Poitiers, Dec. 2007).
Patrick Cousot gave an invited talk [20].

Neil D. JONES tribute workshop (Københavns, Denmark, 2007).
Patrick Cousot and Radhia Cousot gave an invited talk at the ribute workshop and festival to honor Professor Dr. Neil D. Jones [43].

SAS: Static Analysis Symposium (Kongens Lyngby, Denmark, Aug. 2007).
Patrick Cousot, Radhia Cousot, and Xavier Rival attended the conference.

SEFM (London, UK, Sep. 2007).
Patrick Cousot gave an invited tutorial at the Fifth IEEE International Conference on Software Engineering and Formal Methods [22].

SOS: Structural Operational Semantics (Wroclaw, Poland, July 2007).
Patrick Cousot presented [24].

Grand Colloque STIC (Cité des Sciences et de l’Industrie, Paris, La Villette, November 2007).
Bruno Blanchet presented a talk on the FORMACRYPT ARA.
Antoine Miné presented (with Bertrand Jeannet [team PopArt, INRIA-RA]) a poster and a demonstration of the APRON library.

TASE: Theoretical Aspects of Software Engineering (Shanghai, China, June 2007).
Patrick Cousot presented [26] and gave two tutorials on the foundations and applications of abstract interpretation [19].

8.3.2. Invitations and Participation in Seminars

Bruno Blanchet presented the CRYPTOVERIF verifier in the seminar “Formal Protocol Verification Applied”, Schloss Dagstuhl, Wadern, Germany, October 2007 [32].

Patrick Cousot gave seminars at the IBM Thomas J. Watson Research Center [38], [42], the Imperial College [35], New York University [21], École Polytechnique Fédérale de Lausanne [39], Laboratoire d'Informatique de Nantes-Atlantique [41]

Jérôme Feret presented an abstract interpretation framework for analyzing biological pathways in a seminar at the École Normale Supérieure and in a working group at Paris VII. He presented an abstract domain for analyzing digital filtering in embedded software in a seminar at Microsoft Research (Redmond).

Antoine Miné presented the APRON library in a seminar at the Commissariat à l'Énergie Atomique.

Xavier Rival presented the ASTRÉE analyzer in a seminar at the University of California at Berkeley (USA) and in a seminar at Sun Microsystems Labs in Menlo Park (USA). He gave a talk on certified compilation at the Commissariat á l'Énergie Atomique.

9. Bibliography

Major publications by the team in recent years

[1] B. BLANCHET. *A Computationally Sound Mechanized Prover for Security Protocols*, in "IEEE Symposium on Security and Privacy, Oakland, California", May 2006, p. 140-154.

[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. *Verification by Abstract Interpretation*, in "Proc. Int. Symp. on Verification – Theory & Practice, Taormina, Italy", N. DERSHOWITZ (editor), © Springer-Verlag, Berlin, Germany, June 29 – July 4 2003, p. 243–268.

[5] P. COUSOT. *Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming*, in "Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05), Paris, France, LNCS 3385", Springer, Berlin, 17–19 January 2005, p. 1–24.

[6] 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.

[7] 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.

[8] P. COUSOT, R. COUSOT. *An Abstract Interpretation-Based Framework for Software Watermarking*, 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, 14–16 January 2004, p. 173–185.

[9] 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.

[10] A. MINÉ. *Field-Sensitive Value Analysis of Embedded C Programs with Union Types and Pointer Arithmetics*, in "Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES'2006", ACM Press, New York, USA, June 2006, p. 54–63.

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

[12] 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

[13] L. MAUBORGNE. *Analyse statique et domaines abstraits symboliques*, Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, Paris, France, February 2007.

[14] É.-J. SIMS. *Pointer Analysis and Separation Logic*, Ph. D. Thesis, École Polytechnique, Palaiseau, France, December 2007.

Articles in refereed journals and book chapters

[15] M. ABADI, B. BLANCHET, C. FOURNET. *Just Fast Keying in the Pi Calculus*, in "ACM Transactions on Information and System Security (TISSEC)", vol. 10, n^{o} 3, July 2007, p. 1–59.

[16] X. RIVAL, L. MAUBORGNE. *The trace partitioning abstract domain*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", vol. 29, n^{o} 5, August 2007.

Publications in Conferences and Workshops

[17] 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.

[18] B.-Y. E. CHANG, X. RIVAL, G. C. NECULA. *Shape Analysis with Structural Invariant Checkers*, in "Proceedings of the Fourteenth International Symposium on Static Analysis, SAS'07, Kongens Lyngby, Denmark", G. FILÉ, H. RIIS-NIELSON (editors), Lecture Notes in Computer Science, vol. 4634, Springer, Berlin, Germany, 22–24 August 2007, p. 384-401.

[19] P. COUSOT. *Abstract Interpretation and Application to Static Analysis, invited tutorial. Part I: Basic Concepts of Abstract Interpretation; Part II: Applications of Abstract Interpretation*, in "First IEEE & IFIP International Symposium on Theoretical Aspects of Software Engineering, TASE'07, Shanghai, China", 5 June 2007.

[20] P. COUSOT. *Avionic Software Verification by Abstract Interpretation, Invited talk*, in "ISoLA 2007 Workshop on Leveraging Applications of Formal Methods, Verification and Validation", Y. AÏT-AMEUR, F. BONIOL, V.

WIELS (editors), Revue des Nouvelles Technologies de l'Information, vol. RNTI-SM-1, Cépaduès éditions, Toulouse, 12–15 December 2007, p. 1.

[21] 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. KIRSCH, R. WILHELM (editors), ACM Press, New York, NY, USA, 2007, p. 7–9.

[22] P. COUSOT. *The Rôle of Abstract Interpretation in Formal Methods, invited tutorial*, in "Proceedings of the Fifth IEEE International Conference on Software Engineering and Formal Methods, SEFM'2007, London UK,", M. HINCHEY, T. MARGARIA (editors), IEEE Computer Society Press, Los Alamitos, California, USA, 10–14 September 2007, p. 135–137.

[23] P. COUSOT. *The Verification Grand Challenge and Abstract Interpretation*, in "Verified Software: Theories, Tools, Experiments", B. MEYER, J. WOODCOCK (editors), Lecture Notes in Computer Science, vol. 4171, Springer, Berlin, Germany, 2007, p. 227–240.

[24] P. COUSOT, R. COUSOT. *Bi-inductive Structural Semantics*, in "Structural Operational Semantics, SOS'07, Wroclaw, Poland", R. VAN GLABBEEK, M. HENNESSY (editors), ENTCS (1)., vol. 192, n^{o} 1, Elsevier B.V., 9 July 2007.

[25] 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. REPS, M. SAGIV, J. BAUER (editors), Lecture Notes in Computer Science, vol. 4444, Springer, Berlin, Germany, 2007.

[26] 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, H. JIFENG, J. SANDERS (editors), IEEE Computer Society Press, Los Alamitos, California, USA, 6–8 June 2007.

[27] P. COUSOT, P. GANTY, J.-F. RASKIN. *Fixpoint-Guided Abstraction Refinements*, in "Proceedings of the Fourteenth International Symposium on Static Analysis, SAS'07, Kongens Lyngby, Denmark", G. FILÉ, H. RIIS-NIELSON (editors), Lecture Notes in Computer Science, vol. 4634, Springer, Berlin, Germany, 22–24 August 2007, p. 333–348.

[28] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. KRIVINE. *Rule-Based Modelling of Cellular Signalling, invited paper*, in "Proceedings of the Eighteenth International Conference on Concurrency Theory, CONCUR'2007, Lisbon, Portugal, Lisbon, Portugal", L. CAIRES, V. VASCONCELOS (editors), Lecture Notes in Computer Science, vol. 4703, Springer, Berlin, Germany, 3–8 September 2007, p. 17–41.

[29] V. DANOS, J. FERET, W. FONTANA, J. KRIVINE. *Scalable Simulation of Cellular Signaling Networks, invited paper*, in "Proceedings of the Fifth Asian Symposium on Programming Systems, APLAS'2007, Singapore, Singapore", Z. SHAO (editor), Lecture Notes in Computer Science, vol. 4807, Springer, Berlin, Germany, 29 November – 1 December 2007, p. 139–157.

[30] J. FERET. *Reachability Analysis of Biological Signalling Pathways by Abstract Interpretation*, in "Proceedings of the International Conference of Computational Methods in Sciences and Engineering, ICCMSE'2007,

Corfu, Greece, Corfu, Greece", T. SIMOS (editor), American Institute of Physics Conference Proceedings, vol. 2, n^{o} 963, American Institute of Physics, 25–30 September 2007, p. 619–622.

[31] D. MONNIAUX. *Verification of device drivers and intelligent controllers: a case study*, in "Proceedings of the seventh ACM & IEEE international conference on Embedded Systems Software (EMSOFT'07), Salzburg, Austria", ACM Press, 1–3 October 2007, p. 30–36.

Miscellaneous

[32] B. BLANCHET. *CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols*, Shloss Dagstuhl seminar "Formal Protocol Verification Applied"Wadern, Germany, October 2007.

[33] B. BLANCHET, A. D. JAGGARD, A. SCEDROV, J.-K. TSAY. *Computationally Sound Mechanized Proofs of Basic and Public-key Kerberos*, Shloss Dagstuhl seminar "Formal Protocol Verification Applied"Wadern, Germany, October 2007.

[34] P. COUSOT. *Abstract interpretation with applications to semantics and static analysis*, CS Colloquium, Departmental Seminar, New York University, New York, New York, USA, 9 April 2007.

[35] P. COUSOT. *Bi-inductive structural semantics and its abstraction*, Departmental Seminar, Department of Computing, Imperial College, London, UK, 4 July 2007.

[36] P. COUSOT. *Course on Abstract Interpretation*, IBM Thomas J. Watson Research Center, Hawthorne, New York, USA, 6 April—11 May 2007.

[37] P. COUSOT. *Foundations of abstract interpretation: application to semantics*, M1 course of the École Normale Supérieure, 2007.

[38] P. COUSOT. *Program termination proofs by convex optimization*, Seminar, IBM Thomas J. Watson Research Center, Hawthorne, New York, USA, 5 January 2007.

[39] P. COUSOT. *Software Verification by Abstract Interpretation*, School of Computer and Communication Sciences Seminar, École Polytechnique Fédérale de Lausanne, Switzerland, 10 December 2007.

[40] P. COUSOT. *Static Analysis and Verification of Synchronous Embedded Code by Abstract Interpretation, invited talk*, Thirteenth ASTReNet Workshop "Formal Aspects of Source Code Analysis and Manipulation", London, UK, 21 March 2007.

[41] P. COUSOT. *Vérification de logiciel embarqués critiques par interprétation abstraite*, Séminaire du Laboratoire d'Informatique de Nantes-Atlantique (LINA), 20 December 2007.

[42] P. COUSOT, R. COUSOT. *Combination of Abstractions in the* ASTRÉE *Static Analyzer*, 7 May 2007.

[43] P. COUSOT, R. COUSOT. *Specification and Abstraction of Semantics*, In A tribute workshop and festival to honor Professor Dr. Neil D. Jones, Datalogisk Institut, Københavns Universitet, Denmark, 25–26 August 2007.

[44] P. COUSOT, R. COUSOT, L. MAUBORGNE, A. MINÉ, X. RIVAL. *Foundations of abstract interpretation: application to semantics*, M2 course of the MPRI (Master Parisien de Recherche en Informatique), 2007.

[45] P. COUSOT, L. MAUBORGNE. *Programming Languages and Compilation*, L3 course of the École Normale Supérieure, 2007.

[46] P. COUSOT, L. MAUBORGNE, A. MINÉ, X. RIVAL. *Training Session on* ASTRÉE, École Normale Supérieure, Paris, 30 October 2007.

[47] J. FERET. *Accessibilité et simplification automatiques de modèles kappa - BNG*, Groupe de travail Concurrence, PPS, Paris, 4 avril 2007.

[48] J. FERET. *Accessibilité et simplification automatiques de modèles kappa - BNG*, Séminaire Sémantique et Interprétation Abstraite, ENS, Paris, 23 Mars 2007.

[49] J. FERET. *How does the ASTREE analyzer deal with digital filters ?*, Seminar, Microsoft Research, Redmond, USA, 16 August 2007.

[50] B. JEANNET, A. MINÉ. *The* APRON *Library for Numerical Abstract Domains*, August 2007, Poster in the Fourteenth International Symposium on Static Analysis, SAS'07, Kongens Lyngby, Denmark.

[51] A. MINÉ. *The* APRON *Library*, Seminar, Commissariat à l'Énergie Atomique, Saclay, France, December 2007.

[52] X. RIVAL. *The Astrée analyzer*, "Open Source Quality Lunch" Seminar, University of California at Berkeley (USA), April 2007.

[53] X. RIVAL. *The Astrée analyzer*, Seminar, Sun Labs, Palo Alto (USA), April 2007.

References in notes

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

[55] 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.

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

[57] 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, New York, United States, 1977, p. 238–252.

[58] 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.

[59] 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.