IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2014

Team ANTIQUE

Analyse Statique par Interprétation Abstraite

RESEARCH CENTER
Paris - Rocquencourt

THEME
Proofs and Verification

Table of contents

Team ANTIQUE

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

The Team is joint with ENS and CNRS and is part of DIENS (UMR 8548). It is located in the École Normale Supérieure, in Paris.

Creation of the Team: 2014 January 01.

1. Members

Research Scientists

Xavier Rival [Team leader, Inria, Researcher, HdR]
Patrick Cousot [ENS Paris, Professor Emeritus, and Full Professor at the Courant Institute of Mathematical Sciences, NYU, USA]
Radhia Cousot [CNRS, Senior Researcher, Emerite, HdR]
Vincent Danos [CNRS, Senior Researcher, since September 2014, HdR]
Jérôme Feret [Inria, Researcher]
Antoine Miné [CNRS, Researcher, HdR]

Engineers

François Berenger [Inria, from July 2014]
Pippijn Van Steenhoven [Inria, until July 2014]

PhD Students

Mehdi Bouaziz [ENS Paris, granted by ENS Paris]
Ferdinanda Camporesi [ENS Paris, granted by ENS Paris]
Tie Cheng [Inria, granted by European Research Council]
Arlen Cox [Inria, granted by European Research Council, until Septemper 2014]
Huisong Li [Inria, granted by European Research Council]
Antoine Toubhans [Inria, granted by Ministère de la Recherche until August 2014, granted by European Research Council since September 2014]
Caterina Urban [ENS Paris, granted by ENS Paris]
Thibault Suzanne [ENS Paris, granted by École doctorale de Sciences Mathématiques de Paris Centre, since September 2014]

Post-Doctoral Fellows

Wassim Abou Jaoudé [ENS Paris, granted by ENS Paris]
Norman Ferns [ENS Paris, granted by ENS Paris]
Arnaud Spiwack [Inria, granted by European Research Council, until August 2014]

Visiting Scientists

Sukyoung Ryu [KAIST, Associate Professor, Visiting Scientist, from Jul 2014 until Aug 2014]
Kwangkeun Yi [SNU, Professor, Visiting Scientist, June 2014]
Abdelraouf Ouadjaout [PhD, CERIST Research Center, Alger, November 2014]

Administrative Assistants

Nathalie Gaudechoux [Inria, since May 2014]
Marine Meyer [Inria, until April 2014]

2. Overall Objectives

2.1. Overall Objectives

Our group focuses on developing *automated* techniques to compute *semantic properties* of programs and other systems with a computational semantics in general. Such properties include (but are not limited to) important classes of correctness properties.

Verifying safety critical systems (such as avionics systems) is an important motivation to compute such properties. Indeed, a fault in an avionics system, such as a runtime error in the fly-by-wire command software, may cause an accident, with loss of life. As these systems are also very complex and are developed by large teams and maintained over long periods, their verification has became a crucial challenge. Safety critical systems are not limited to avionics: software runtime errors in cruise control management systems were recently blamed for causing *unintended acceleration* in certain Toyota models (the case was settled with a 1.2 billion dollars fine in March 2014, after years of investigation and several trials). Similarly, other transportation systems (railway), energy production systems (nuclear power plants, power grid management), and medical systems (pacemakers, surgery and patient monitoring systems) rely on complex software, which should be verified.

Beyond the field of embedded systems, other pieces of software may cause very significant harm in case of bugs, as demonstrated by the Heartbleed security hole: due to a wrong protocol implementation, many websites could leak private information, over years.

An important example of semantic properties is the class of *safety* properties. A safety property typically specifies that some (undesirable) event will never occur, whatever the execution of the program that is considered. For instance, the absence of runtime error is a very important safety property. Other important classes of semantic properties include *liveness* properties (i.e., properties that specify that some desirable event will eventually occur) such as termination and *security* properties, such as the absence of information flows from private to public channels.

All these software semantic properties are *not decidable*, as can be shown by reduction to the halting problem. Therefore, there is no chance to develop any fully automatic technique able to decide, for any system, whether or not it satisfies some given semantic property.

The classic development techniques used in industry involve testing, which is not sound, as it only gives information about a usually limited test sample: even after successful test-based validation, situations that were untested may generate a problem. Furthermore, testing is costly in the long term, as it should be re-done whenever the system to verify is modified. Machine-assisted verification is another approach which verifies human specified properties. However, this approach also presents a very significant cost, as the annotations required to verify large industrial applications would be huge.

By contrast, the **antique** group focuses on the design of semantic analysis techniques that should be *sound* (i.e., compute semantic properties that are satisfied by all executions) and *automatic* (i.e., with no human interaction), although generally *incomplete* (i.e., not able to compute the best —in the sense of: most precise— semantic property). As a consequence of incompleteness, we may fail to verify a system that is actually correct. For instance, in the case of verification of absence of runtime error, the analysis may fail to validate a program, which is safe, and emit *false alarms* (that is reports that possibly dangerous operations were not proved safe), which need to be discharged manually. Even in this case, the analysis provides information about the alarm context, which may help disprove it manually or refine the analysis.

The methods developed by the **antique** group are not be limited to the analysis of software. We also consider complex biological systems (such as models of signaling pathways, i.e. cascades of protein interactions, which enable signal communication among and within cells), described in higher level languages, and use abstraction techniques to reduce their combinatorial complexity and capture key properties so as to get a better insight in the underlying mechanisms of these systems.

3. Research Program

3.1. Semantics

Semantics plays a central role in verification since it always serves as a basis to express the properties of interest, that need to be verified, but also additional properties, required to prove the properties of interest, or which may make the design of static analysis easier.

For instance, if we aim for a static analysis that should prove the absence of runtime error in some class of programs, the concrete semantics should define properly what error states and non error states are, and how program executions step from a state to the next one. In the case of a language like C, this includes the behavior of floating point operations as defined in the IEEE 754 standard. When considering parallel programs, this includes a model of the scheduler, and a formalization of the memory model.

In addition to the properties that are required to express the proof of the property of interest, it may also be desirable that semantics describe program behaviors in a finer manner, so as to make static analyses easier to design. For instance, it is well known that, when a state property (such as the absence of runtime error) is valid, it can be established using only a state invariant (i.e., an invariant that ignores the order in which states are visited during program executions). Yet searching for trace invariants (i.e., that take into account some properties of program execution history) may make the static analysis significantly easier, as it will allow it to make finer case splits, directed by the history of program executions. To allow for such powerful static analyses, we often resort to a *non standard semantics*, which incorporates properties that would normally be left out of the concrete semantics.

3.2. Abstract interpretation and static analysis

Once a reference semantics has been fixed and a property of interest has been formalized, the definition of a static analysis requires the choice of an *abstraction*. The abstraction ties a set of *abstract predicates* to the concrete ones, which they denote. This relation is often expressed with a *concretization function* that maps each abstract element to the concrete property it stands for. Obviously, a well chosen abstraction should allow expressing the property of interest, as well as all the intermediate properties that are required in order to prove it (otherwise, the analysis would have no chance to achieve a successful verification). It should also lend itself to an efficient implementation, with efficient data-structures and algorithms for the representation and the manipulation of abstract predicates. A great number of abstractions have been proposed for all kinds of concrete data types, yet the search for new abstractions is a very important topic in static analysis, so as to target novel kinds of properties, to design more efficient or more precise static analyses.

Once an abstraction is chosen, a set of *sound abstract transformers* can be derived from the concrete semantics and that account for individual program steps, in the abstract level and without forgetting any concrete behavior. A static analysis follows as a result of this step by step approximation of the concrete semantics, when the abstract transformers are all computable. This process defines an *abstract interpretation* [40]. The case of loops requires a bit more work as the concrete semantics typically relies on a fixpoint that may not be computable in finitely many iterations. To achieve a terminating analysis we then use *widening operators* [40], which over-approximates the concrete union and ensure termination.

A static analysis defined that way always terminates and produces sound over-approximations of the programs behaviors. Yet, these results may not be precise enough for verification. This is where the art of static analysis design comes into play through, among others:

- the use of more precise, yet still efficient enough abstract domains;
- the combination of application specific abstract domains;
- the careful choice of abstract transformers and widening operators.

3.3. Applications of the notion of abstraction in semantics

In the previous subsections, we sketched the steps in the design of a static analyzer to infer some family of properties, which should be implementable, and efficient enough to succeed in verifying non trivial systems.

Yet, the same principles can also be applied successfully to other goals. In particular, the abstract interpretation framework should be viewed a very general tool to *compare different semantics*, not necessarily with the goal of deriving a static analyzer. Such comparisons may be used in order to prove two semantics equivalent (i.e., one is an abstraction of the other and vice versa), or that a first semantics is strictly more expressive than another one (i.e., the latter can be viewed an abstraction of the former, where the abstraction actually makes some information redundant, which cannot be recovered). A classical example of such comparison is the classification of semantics of transition systems [38], which provides a better understanding of program semantics in general. For instance, this approach can be applied to get a better understanding of the semantics of a programming language, but also to select which concrete semantics should be used as a foundation for a static analysis, or to prove the correctness of a program transformation, compilation or optimization.

3.4. The analysis of biological models

One of our application domains, the analysis of biological models, is not a classical target of static analysis because it aims at analyzing models instead of programs. Yet, the analysis of biological models is closely intertwined with the other application fields of our group. Firstly, abstract interpretation provides a formal understanding of the abstraction process which is inherent to the modeling process. Abstract interpretation is also used to better understand the systematic approaches which are used in the systems biology field to capture the properties of models, until getting formal, fully automatic, and scalable methods. Secondly, abstract interpretation is used to offer various semantics with different grains of abstraction, and, thus, new methods to apprehend the overall behavior of the models. Conversely, some of the methods and abstractions which are developed for biological models are inspired by the analysis of concurrent systems and by security analysis. Lastly, the analysis of biological models raises issues about differential systems, stochastic systems, and hybrid systems. Any breakthrough in these directions will likely be very important to address the important challenge of the certification of critical systems in interaction with their physical environment.

4. Application Domains

4.1. Verification of safety critical embedded software

The verification of safety critical embedded software is a very important application domain for our group. First, this field requires a high confidence in software, as a bug may cause disastrous events. Thus, it offers an obvious opportunity for a strong impact. Second, such software usually have better specifications and a better design than many other families of software, hence are an easier target for developing new static analysis techniques (which can later be extended for more general, harder to cope with families of programs). This includes avionics, automotive and other transportation systems, medical systems...

For instance, the verification of avionics systems represent a very high percentage of the cost of an airplane (about 30 % of the overall airplane design cost). The state of the art development processes mainly resort to testing in order to improve the quality of software. Depending on the level of criticality of a software (at highest levels, any software failure would endanger the flight) a set of software requirements are checked with test suites. This approach is both costly (due to the sheer amount of testing that needs to be performed) and unsound (as errors may go unnoticed, if they do not arise on the test suite).

By contrast, static analysis can ensure higher software quality at a lower cost. Indeed, a static analyzer will catch all bugs of a certain kind. Moreover, a static analysis run typically lasts a few hours, and can be integrated in the development cycle in a seamless manner. For instance, ASTRÉEsuccessfully verified the absence of runtime error in several families of safety critical fly-by-wire avionic software, in at most a day of computation, on standard hardware. Other kinds of synchronous embedded software have also been analyzed with good results.

In the future, we plan to greatly extend this work so as to verify *other families of embedded software* (such as communication, navigation and monitoring software) and *other families of properties* (such as security and liveness properties).

Embedded software in charge of communication, navigation, monitoring typically rely on a *parallel* structure, where several threads are executed in parallel, and manage different features (input, output, user interface, internal computation, logging...). This structure is also often found in automotive software. An even more complex case is that of *distributed* systems, where several separate computers are run in parallel and take care of several sub-tasks of a same feature, such as braking. Such a logical structure is not only more complex than the synchronous one, but it also introduces new risks and new families of errors (deadlocks, data-races...). Moreover, such less well designed, and more complex embedded software often utilizes more complex data-structures than synchronous programs (which typically only use arrays to store previous states) and may use dynamic memory allocation, or build dynamic structures inside static memory regions, which are actually even harder to verify than conventional dynamically allocated data structures. Complex data-structures also introduce new kinds of risks (the failure to maintain structural invariants may lead to runtime errors, non termination, or other software failures). To verify such programs, we will design additional abstract domains, and develop new static analysis techniques, in order to support the analysis of more complex programming language features such as parallel and concurrent programming with threads and manipulations of complex data structures. Due to their size and complexity, the verification of such families of embedded software is a major challenge for the research community.

Furthermore, embedded systems also give rise to novel security concerns. It is in particular the case for some aircraft-embedded computer systems, which communicate with the ground through untrusted communication media. Besides, the increasing demand for new capabilities, such as enhanced on-board connectivity, e.g. using mobile devices, together with the need for cost reduction, leads to more integrated and interconnected systems. For instance, modern aircrafts embed a large number of computer systems, from safety-critical cockpit avionics to passenger entertainment. Some systems meet both safety and security requirements. Despite thorough segregation of subsystems and networks, some shared communication resources raise the concern of possible intrusions. Because of the size of such systems, and considering that they are evolving entities, the only economically viable alternative is to perform automatic analyses. Such analyses of security and confidentiality properties have never been achieved on large-scale systems where security properties interact with other software properties, and even the mapping between high-level models of the systems and the large software base implementing them has never been done and represents a great challenge. Our goal is to prove empirically that the security of such large scale systems can be proved formally, thanks to the design of dedicated abstract interpreters.

The long term goal is to make static analysis more widely applicable to the verification of industrial software.

4.2. Static analysis of software components and libraries

An important goal of our work is to make static analysis techniques easier to apply to wider families of software. Then, in the longer term, we hope to be able to verify less critical, yet very commonly used pieces of software. Those are typically harder to analyze than critical software, as their development process tends to be less rigorous. In particular, we will target operating systems components and libraries. As of today, the verification of such programs is considered a major challenge to the static analysis community.

As an example, most programming languages offer Application Programming Interfaces (API) providing ready-to-use abstract data structures (e.g., sets, maps, stacks, queues, etc.). These APIs, are known under the name of containers or collections, and provide off-the-shelf libraries of high level operations, such as insertion, deletion and membership checks. These container libraries give software developers a way of abstracting from low-level implementation details related to memory management, such as dynamic allocation, deletion and pointer handling or concurrency aspects, such as thread synchronization. Libraries implementing data structures are important building bricks of a huge number of applications, therefore their verification is paramount. We are interested in developing static analysis techniques that will prove automatically the correctness of large audience libraries such as Glib and Threading Building Blocks.

4.3. Biological systems

Computer Science takes a more and more important role in the design and the understanding of biological systems such as signaling pathways, self assembly systems, DNA repair mechanisms. Biology has gathered large data-bases of facts about mechanistic interactions between proteins, but struggles to draw an overall picture of how these systems work as a whole. High level languages designed in Computer Science allow to collect these interactions in integrative models, and provide formal definitions (i.e., semantics) for the behavior of these models. This way, modelers can encode their knowledge, following a bottom-up discipline, without simplifying *a priori* the models at the risk of damaging the key properties of the system. Yet, the systems that are obtained this way suffer from combinatorial explosion (in particular, in the number of different kinds of molecular components, which can arise at run-time), which prevents from a naive computation of their behavior.

We develop various abstract interpretation-based analyses, tailored to different phases of the modeling process. We propose automatic static analyses in order to detect inconsistencies in the early phases of the modeling process. These analyses are similar to the analysis of classical safety properties of programs. They involve both forward and backward reachability analyses as well as causality analyses, and can be tuned at different levels of abstraction. We also develop automatic static analyses so as to identify the key elements in the dynamics of these models. The results of these analyses are sent to another tool, which is used to automatically simplify the models. The correctness of this simplification process is proved by the means of abstract interpretation: this ensures formally that the simplification preserves the quantitative properties that have been specified beforehand by the modeler. The whole pipeline is parameterized by a large choice of abstract domains which exploits different features of the high level description of models.

5. New Software and Platforms

5.1. The Apron Numerical Abstract Domain Library

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

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

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

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

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

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

Additionally, APRON is used internally by the team to assist the research on numeric domains and static analyses by enabling the development of fast prototypes. Specifically, in 2014, APRON has been used to support the design of piece-wise linear ranking function domains to infer termination and functional liveness properties in the implementation of the FUNCTION prototype analyzer, and to implement and experiment a new numeric domain for octagonal constraints with absolute values. It has also been used in the introductory course on program verification given by members of the team.

5.2. The Astrée Static Analyzer of Synchronous Software

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

ASTRÉE is a static analyzer for sequential programs based on abstract interpretation [40], [35], [41], [36].

The ASTRÉE static analyzer [34], [44][1] www.astree.ens.fr aims at proving the absence of runtime errors in programs written in the C programming language.

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

ASTRÉE discovers all runtime errors including:

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

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

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

ASTRÉE has achieved the following two unprecedented results:

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

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

5.3. The AstréeA Static Analyzer of Asynchronous Software

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

ASTRÉEA is a static analyzer prototype for parallel software based on abstract interpretation [42], [43], [37]. It started with support from THÉSÉE ANR project (2006–2010) and is continuing within the ASTRÉEA project (2012–2015).

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

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

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

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

5.4. ClangML: A binding with the CLANG C-frontend

Participants: François Berenger [Correspondent], Devin Mccoughlin, Pippijn Van Steenhoeven.

CLANGML is an OCaml binding with the CLANG front-end of the LLVM compiler suite. Its goal is to provide an easy to use solution to parse a wide range of C programs, that can be called from static analysis tools implemented in OCaml, which allows to test them on existing programs written in C (or in other idioms derived from C) without having to redesign a front-end from scratch. CLANGML features an interface to a large set of internal AST nodes of CLANG, with an easy to use API. Currently, CLANGML supports all C language AST nodes, as well as a large part of the C nodes related to C++ and Objective-C.

It has been applied to the parsing of the Minix micro-kernel as well as of other C programs.

CLANGML has been implemented in C++, OCaml and Camlp4. It has been released as an open source contribution on GitHUB and as an OPAM package.

5.5. FuncTion: An Abstract Domain Functor for Termination

Participants: Caterina Urban, Antoine Miné [Correspondent].

FUNCTION is a research prototype static analyzer to analyze the termination and functional liveness properties of programs. It accepts programs in a small non-deterministic imperative language. It is also parameterized by a property: either termination, or a recurrence or a guarantee property (according to the classification by Manna and Pnueli of program properties). It then performs a backward static analysis that automatically infers sufficient conditions at the beginning of the program so that all executions satisfying the conditions also satisfy the property. FUNCTION is based on an extension to liveness properties of the framework to analyze termination by abstract interpretation proposed by Patrick Cousot and Radhia Cousot in [39]. FUNCTION infers ranking functions using piecewise-defined abstract domains. Several domains are available to partition the ranking function, including intervals, octagons, and polyhedra. Two domains are also available to represent the value of ranking functions: a domain of affine ranking functions, and a domain of ordinal-valued ranking functions (which allows handling programs with unbounded non-determinism).

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

FUNCTION participated to SV-COMP 2014 (3rd Competition on Software Verification, demonstration section) and is also selected to participate to SV-COMP 2015 in the termination category [31].

5.6. HOO: Heap Abstraction for Open Objects

Participant: Arlen Cox [Correspondent].

JSAna with HOO is a static analyzer for JavaScript programs. The primary component, HOO, which is designed to be reusable by itself, is an abstract domain for a dynamic language heap. A dynamic language heap consists of open, extensible objects linked together by pointers. Uniquely, HOO abstracts these extensible objects, where attribute/field names of objects may be unknown. Additionally, it contains features to keeping precise track of attribute name/value relationships as well as calling unknown functions through desynchronized separation.

As a library, HOO is useful for any dynamic language static analysis. It is designed to allow abstractions for values to be easily swapped out for different abstractions, allowing it to be used for a wide-range of dynamic languages outside of JavaScript.

5.7. The MemCADstatic analyzer

Participants: Xavier Rival [correspondent], François Berenger, Huisong Li, Antoine Toubhans.

Shape analysis. MEMCADis a static analyzer that focuses on memory abstraction. It takes as input C programs, and computes invariants on the data structures manipulated by the programs. It can also verify memory safety. It comprises several memory abstract domains, including a flat representation, and two graph abstractions with summaries based on inductive definitions of data-structures, such as lists and trees and several combination operators for memory abstract domains (hierarchical abstraction, reduced product). The purpose of this construction is to offer a great flexibility in the memory abstraction, so as to either make very efficient static analyses of relatively simple programs, or still quite efficient static analyses of very involved pieces of code. The implementation consists of over 30 000 lines of ML code, and relies on the CLANGML front-end. The current implementation comes with over 300 small size test cases that are used as regression tests.

5.8. The OpenKappa Modeling Plateform

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

Causal traces, Model reduction, Rule-based modeling, Simulation, Static analysis. OPENKAPPA is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language [50], a static analyzer [49] (for debugging models), a simulator [48], a compression tool for causal traces [47], [45], and a model reduction tool [4], [46], [53].

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

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

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

5.9. QUICr set abstract domain

Participant: Arlen Cox [Correspondent].

QUICr is an OCaml library that implements a parametric abstract domain for sets. It is constructed as a functor that accepts any numeric abstract domain that can be adapted to the interface and produces an abstract domain for sets of numbers combined with numbers. It is relational, flexible, and tunable. It serves as a basis for future exploration of set abstraction.

5.10. Translation Validation

Participant: Xavier Rival [correspondent].

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

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

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

5.11. Zarith

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

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

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

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

6. New Results

6.1. Highlights of the Year

Patrick and Radhia Cousot have received in 2014 the IEEE Computer Society IEEE Computer Society Harlan D. Mills award for the invention of abstract interpretation, developpment of tool support and practical application http://www.computer.org/portal/web/awards/cousots.

6.2. Memory Abstraction

6.2.1. Modular Construction of Shape-Numeric Analyzers

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

Abstract interpretation, Memory abstraction, Shape abstract domains. In [24], we discuss the modular construction of memory abstract domains.

The aim of static analysis is to infer invariants about programs that are tight enough to establish semantic properties, like the absence of run-time errors. In the last decades, several branches of the static analysis of imperative programs have made significant progress, such as in the inference of numeric invariants or the computation of data structures properties (using pointer abstractions or shape analyzers). Although simultaneous inference of shape-numeric invariants is often needed, this case is especially challenging and less well explored. Notably, simultaneous shape-numeric inference raises complex issues in the design of the static analyzer itself. We studied the modular construction of static analyzers, based on combinations of atomic abstract domains to describe several kinds of memory properties and value properties.

6.2.2. An abstract domain combinator for separately conjoining memory abstractions

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

Abstract interpretation, Memory abstraction, Shape abstract domains. In [25], we studied the separating combination of heap abstract domains.

The breadth and depth of heap properties that can be inferred by the union of today's shape analyses is quite astounding. Yet, achieving scalability while supporting a wide range of complex data structures in a generic way remains a long-standing challenge. We proposed a way to side-step this issue by defining a generic abstract domain combinator for combining memory abstractions on disjoint regions. In essence, our abstract domain construction is to the separating conjunction in separation logic as the reduced product construction is to classical, non-separating conjunction. This approach eases the design of the analysis as memory abstract domains can be re-used by applying our separating conjunction domain combinator. And more importantly, this combinator enables an analysis designer to easily create a combined domain that applies computationally-expensive abstract domains only where it is required.

6.2.3. Abstraction of Arrays Based on Non Contiguous Partitions

Participants: Xavier Rival [correspondant], Jiangchao Liu.

Abstract interpretation, Memory abstraction, Array abstract domains. In [20], we studied array abstractions.

Array partitioning analyses split arrays into contiguous partitions to infer properties of cell sets. Such analyses cannot group together non contiguous cells, even when they have similar properties. We proposed an abstract domain which utilizes semantic properties to split array cells into groups. Cells with similar properties will be packed into groups and abstracted together. Additionally, groups are not necessarily contiguous. This abstract domain allows to infer complex array invariants in a fully automatic way. Experiments on examples from the Minix 1.1 memory management demonstrated its effectiveness.

6.3. Static analysis of JavaScript applications

6.3.1. Automatic Analysis of Open Objects in Dynamic Language Programs

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

Abstract interpretation, Dynamically typed languages, Verification In [14], we have studied the abstraction of open objects in dynamic language programs (like JavaScript).

In dynamic languages, objects are open: they support iteration over and dynamic addition/deletion of their attributes. Open objects, because they have an unbounded number of attributes, are difficult to abstract without a priori knowledge of all or nearly all of the attributes and thus pose a significant challenge for precise static analysis. To address this challenge, we presented the HOO (Heap with Open Objects) abstraction that can precisely represent and infer properties about open-object-manipulating programs without any knowledge of specific attributes. It achieves this by building upon a relational abstract domain for sets that is used to reason about partitions of object attributes. An implementation of the resulting static analysis is used to verify specifications for dynamic language framework code that makes extensive use of open objects, thus demonstrating the effectiveness of this approach.

6.3.2. Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages

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

Abstract interpretation, Dynamically typed languages, Verification In [15], we have studied desynchronized multi-state abstractions for open programs in dynamic languages (libraries).

Dynamic language library developers face a challenging problem: ensuring that their libraries will behave correctly for a wide variety of client programs without having access to those client programs. This problem stems from the common use of two defining features for dynamic languages: callbacks into client code and complex manipulation of attribute names within objects. To remedy this problem, we introduced two state-spanning abstractions. To analyze callbacks, the first abstraction desynchronizes a heap, allowing partitions of the heap that may be affected by a callback to an unknown function to be frozen in the state prior to the call. To analyze object attribute manipulation, building upon an abstraction for dynamic language heaps, the second abstraction tracks attribute name/value pairs across the execution of a library. We implemented these abstractions and use them to verify modular specifications of class-, trait-, and mixin-implementing libraries.

6.4. Static analysis of Spreadsheet applications

Participants: Tie Cheng [correspondant], Xavier Rival.

Abstract interpretation, Spreadsheet applications, Verification In [13], we have proposed a static analysis to detect type unsafe operations in spreadsheet applications including formulas and macros.

Spreadsheets are widely used, yet are error-prone: they use a weak type system, allowing certain operations that will silently return unexpected results, like comparisons of integer values with string values. However, discovering these issues is hard, since data and formulas can be dynamically set, read or modified. We defined a static analysis that detects all run-time type-unsafe operations in spreadsheets. It is based on an abstract interpretation of spreadsheet applications, including spreadsheet tables, global re-evaluation and associated programs. Our implementation supports the features commonly found in real-world spreadsheets. We ran our analyzer on the EUSES Spreadsheet Corpus. This evaluation shows that our tool is able to automatically verify a large number of real spreadsheets, runs in a reasonable time and discovers complex bugs that are difficult to detect by code review or by testing.

6.5. Mechanically Verifying a Shape Analysis

Participant: Arnaud Spiwack.

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

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

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

6.6. Static Analysis of Embedded Critical Concurrent Software

6.6.1. AstréeA: A Static Analyzer for Large Embedded Multi-Task Software

Participant: Antoine Miné.

In [11], we present the design, implementation and experimentation of the AstréeA static analyzer, an extension of the Astrée static analyzer dedicated to analyzing the run-time errors in embedded critical concurrent software. Such software are already present in critical systems and will likely become the norm with the generalization of multi-core processors in embedded systems, leading to new challenging demands in verification. One major challenge is that a concurrent program execution does not follow a fixed sequential order, but one of many interleavings of executions from different tasks chosen by the scheduler. As it is impractical to build a fully flow-sensitive analysis by enumerating explicitly all interleavings, we took inspiration from thread-modular methods: we analyze each thread individually, in an environment consisting of (an abstraction of) the effect of the other threads. This is a form of rely-guarantee reasoning, but in a fully automatic static analysis settings formalized as abstract interpretation: a thread-modular static analysis is viewed as a computable abstraction of a complete concrete, fixpoint-based thread-modular semantics. This permits a fine control between precision and efficiency, and opens the way to analysis specialization: any given safety property of a given program can be theoretically inferred given the right abstract domain. The presentation describes our subsequent work in improving the precision of AstréeA by specialization on our target applications, and the interesting abstractions we developed along the way. For instance, we developed new interference abstractions enabling a limited but controllable (for efficiency) degree of relationality and flow-sensitivity. We also designed abstractions able to exploit our knowledge of the real-time scheduler used in the analysis target: i.e., it schedules tasks on a single core and obeys a strict priority scheme. The end-result is a more precise analyzer on our target applications, with currently around a thousand alarms.

6.6.2. Static Analysis by Abstract Interpretation of Concurrent Programs under the TSO Weak Memory Model

Participants: Thibault Suzanne, Antoine Miné.

In [33], we present an abstract semantics for the Total Store Ordering (TSO) memory model, a weakly consistent memory model used in major multi-core processors. This abstraction forgets some information about the order in which variables are written into by each thread. This results in a much simplified concrete semantics, but which is still not computable. We then express the semantics based on partitioned sets of points in a vector space, which allows applying classic methods from abstract interpretation (such as numeric abstract domains) to achieve a fully computable abstract semantics and automatically infer an over-approximation of the set of reachable states of a program running under the TSO memory model. The method is proved correct and, in certain cases, optimal, using the standard tools of abstraction interpretation (Galois connections). Moreover, we have written a prototype static analyzer for simple program fragments written in an assembly-like language, and experimented our abstraction on a few small examples.

6.7. Inference of Termination and Liveness properties

6.7.1. A Decision Tree Abstract Domain for Proving Conditional Termination

Participants: Caterina Urban, Antoine Miné.

In [26], we present a new parameterized abstract domain able to refine existing numerical abstract domains with finite disjunctions. The elements of the abstract domain are decision trees where the decision nodes are labeled with linear constraints, and the leaf nodes belong to a numerical abstract domain. The abstract domain is parametric in the choice between the expressivity and the cost of the linear constraints for the decision nodes (e.g., polyhedral or octagonal constraints), and the choice of the abstract domain for the leaf nodes. We describe an instance of this domain based on piecewise-defined ranking functions for the automatic inference of sufficient preconditions for program termination. We have implemented a static analyzer for proving conditional termination of programs written in (a subset of) C and, using experimental evidence, we show that it performs well on a wide variety of benchmarks, it is competitive with the state of the art and is able to analyze programs that are out of the reach of existing methods.

6.7.2. An Abstract Domain to Infer Ordinal-Valued Ranking Functions

Participants: Caterina Urban, Antoine Miné.

The traditional method for proving program termination consists in inferring a ranking function. In many cases (i.e. programs with unbounded non-determinism), a single ranking function over natural numbers is not sufficient. In [30], we propose a new abstract domain to automatically infer ranking functions over ordinals. We extend an existing domain for piecewise-defined natural-valued ranking functions to polynomials in ω, where the polynomial coefficients are natural-valued functions of the program variables. The abstract domain is parametric in the choice of the state partitioning inducing the piecewise-definition and the type of functions used as polynomial coefficients. To our knowledge this is the first abstract domain able to reason about ordinals. Handling ordinals leads to a powerful approach for proving termination of imperative programs, which in particular allows us to take a first step in the direction of proving termination under fairness constraints and proving liveness properties of (sequential and) concurrent programs.

6.7.3. Proving Guarantee and Recurrence Temporal Properties by Abstract Interpretation

Participants: Caterina Urban, Antoine Miné.

We present in [28] a new static analysis methods for proving liveness properties of programs. In particular, with reference to the hierarchy of temporal properties proposed by Manna and Pnueli, we focus on guarantee (i.e., "something good occurs at least once") and recurrence (i.e., "something good occurs infinitely often") temporal properties. We generalize the abstract interpretation framework for termination presented by Cousot and Cousot. Specifically, static analyses of guarantee and recurrence temporal properties are systematically derived by abstraction of the program operational trace semantics. These methods automatically infer sufficient preconditions for the temporal properties by reusing existing numerical abstract domains based on piecewise-defined ranking functions. We augment these abstract domains with new abstract operators, including a dual widening. To illustrate the potential of the proposed methods, we have implemented a research prototype static analyzer, for programs written in a C-like syntax, that yielded interesting preliminary results.

6.8. Numeric Invariant Inference

6.8.1. A Numeric Abstract Domain to Infer Octagonal Constraints with Absolute Value

Participants: Liqian Chen [National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha, P.R.China], Jiangchao Liu, Antoine Miné, Deepak Kapur [University of New Mexico, USA], Ji Wang [National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha, P.R.China].

The octagon abstract domain, devoted to discovering octagonal constraints (also called Unit Two Variable Per Inequality or UTVPI constraints) of a program, is one of the most commonly used numerical abstractions in practice, due to its quadratic memory complexity and cubic time complexity. However, the octagon domain itself is restricted to express convex sets and has limitations in handling non-convex properties which are sometimes required for proving some numerical properties in a program. In [12], we intend to extend the octagon abstract domain with absolute value, to infer certain non-convex properties by exploiting the absolute value function. More precisely, the new domain can infer relations of the form $\{ \pm X \pm Y \leq c,\ \pm X \pm |Y| \leq d,\ \pm|X| \pm |Y| \leq e \}$. We provide algorithms for domain operations such that the new domain still enjoys the same asymptotic complexity as the octagon domain. Moreover, we present an approach to support strict inequalities over rational or real-valued variables in this domain, which also fits for the octagon domain. Experimental results of our prototype are encouraging; The new domain is scalable and able to find non-convex invariants of interest in practice but without too much overhead (compared with that using octagons).

6.8.2. A Method to Infer Inductive Numeric Invariants Inspired from Constraint Programming.

Participant: Antoine Miné.

In [29], we suggest the idea of using algorithms inspired by Constraint Programming in order to infer inductive invariants on numeric programs. Similarly to Constraint Programming solvers on continuous domains, our algorithm approximates the problem from above, using decreasing iterations that may split, discard, and tighten axis-aligned boxes. If successful, the algorithm outputs a set of boxes that includes the initial states and is a post-fixpoint of the abstract semantic function of interest. Our work is very preliminary; many improvements still need to be performed to determine if the method is usable in practice, and in which contexts. Nevertheless, we show that a naive proof-of-concept implementation of our algorithm is already capable of inferring non-trivial inductive invariants that would otherwise require the use of relational or even non-linear abstract domains when using more traditional abstract interpretation iteration methods.

6.9. Bisimulation Metrics

6.9.1. Bisimulation for Markov Decision Processes through Families of Functional Expressions

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

Markov decision processes, Bisimulation, Metrics.

In [17], we have transfered a notion of quantitative bisimilarity for labelled Markov processes [51] to Markov decision processes with continuous state spaces. This notion takes the form of a pseudometric on the system states, cast in terms of the equivalence of a family of functional expressions evaluated on those states and interpreted as a real-valued modal logic. Our proof amounts to a slight modification of previous techniques [56], [55] used to prove equivalence with a fixed-point pseudometric on the state-space of a labelled Markov process and making heavy use of the Kantorovich probability metric. Indeed, we again demonstrate equivalence with a fixed-point pseudometric defined on Markov decision processes [52]; what is novel is that we recast this proof in terms of integral probability metrics [54] defined through the family of functional expressions, shifting emphasis back to properties of such families. The hope is that a judicious choice of family might lead to something more computationally tractable than bisimilarity whilst maintaining its pleasing theoretical guarantees. Moreover, we use a trick from descriptive set theory to extend our results to MDPs with bounded measurable reward functions, dropping a previous continuity constraint on rewards and Markov kernels.

6.9.2. Bisimulation Metrics are Optimal Value Functions

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

Markov decision processes, Bisimulation, Metrics.

Bisimulation is a notion of behavioural equivalence on the states of a transition system. Its definition has been extended to Markov decision processes, where it can be used to aggregate states. A bisimulation metric is a quantitative analog of bisimulation that measures how similar states are from a the perspective of long-term behavior. Bisimulation metrics have been used to establish approximation bounds for state aggregation and other forms of value function approximation. In [18], we prove that a bisimulation metric defined on the state space of a Markov decision process is the optimal value function of an optimal coupling of two copies of the original model. We prove the result in the general case of continuous state spaces. This result has important implications in understanding the complexity of computing such metrics, and opens up the possibility of more efficient computational methods.

6.10. Abstraction of Rule-Based Biological Models

6.10.1. Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models

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

Protein-protein interaction networks, Stochastic systems, Backward bisimulations, Model reduction. In [9], we propose 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 introduce an individual-based stochastic semantics (where each agent is identified by a unique identifier) for the programs of a rule-based language (namely Kappa) and we use our abstraction framework for deriving 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 [9].

6.10.2. An algebraic approach for inferring and using symmetries in rule-based models

Participant: Jérôme Feret.

Graph rewriting, Single-pushout semantics, Symmetries, Bisimulations, Model reduction. Symmetries arise naturally in rule-based models, and under various forms. Besides automorphisms between site graphs, which are usually built within the semantics, symmetries can take the form of pairs of sites having the same capabilities of interactions, of some protein variants behaving exactly the same way, or of some linear, planar, or 3D molecular complexes which could be seen modulo permutations of their axis and/or mirror-image symmetries. In [16], we propose a unifying handling of symmetries in Kappa. We follow an algebraic approach, that is based on the single pushout semantics of Kappa. We model classes of symmetries as finite groups of transformations between site graphs, which are compatible with the notion of embedding (that is to say that it is always possible to restrict a symmetry that is applied with the image of an embedding to the domain of this embedding) and we provide some assumptions that ensure that symmetries are compatible with pushouts. Then, we characterise when a set of rules is symmetric with respect to a group of symmetries and, in such a case, we give sufficient conditions so that this group of symmetries induces a forward bisimulation and/or a backward bisimulation over the population semantics.

6.11. Model checking of Logical Biological Models

6.11.1. Model checking logical regulatory networks

Participants: Pedro T. Monteiro [INESC-ID, Lisboa, Portugal], Wassim Abou-Jaoudé, Denis Thieffry [IBENS, France], Claudine Chaouiya [IGC, Oeiras, Portugal].

Model checking, Regulatory networks. Regulatory and signalling networks control cell behaviours in response to environmental cues. The logical formalism has been widely employed to study these interaction networks, which are modelled as discrete dynamical systems. While biologists identify networks encompassing more and more components, properties of biological relevance become hard to verify.

In [22], we report on the use of model-checking techniques to address this challenge. This approach is illustrated by an application dealing with the modelling of T-helper lymphocyte differentiation.

6.11.2. Model checking to assess T-helper cell plasticity

Participants: Wassim Abou-Jaoudé, Pedro T. Monteiro [INESC-ID, Lisboa, Portugal], Aurélien Naldi [Centre Intégratif de Lausanne, Lausanne, Switzerland], Maximilien Grandclaudon [Institut Curie, Paris, France], Vassili Sommeils [Institut Curie, Paris, France], Claudine Chaouiya [IGC, Oeiras, Portugal], Denis Thieffry [IBENS, France].

Model checking, Logical modeling. Computational modeling constitutes a crucial step toward the functional understanding of complex cellular networks. In particular, logical modeling has proven suitable for the dynamical analysis of large signaling and transcriptional regulatory networks. In this context, signaling input components are generally meant to convey external stimuli, or environmental cues. In response to such external signals, cells acquire specific gene expression patterns modeled in terms of attractors (e.g., stable states).The capacity for cells to alter or reprogram their differentiated states upon changes in environmental conditions is referred to as cell plasticity. In this article, we present a multivalued logical framework along with computational methods recently developed to efficiently analyze large models.We mainly focus on a symbolic model checking approach to investigate switches between attractors subsequent to changes of input conditions. As a case study, we consider the cellular network regulating the differentiation of T-helper (Th) cells, which orchestrate many physiological and pathological immune responses. To account for novel cellular subtypes, we present, in [8], an extended version of a published model of Th cell differentiation.We then use symbolic model checking to analyze reachability properties between Th subtypes upon changes of environmental cues. This allows for the construction of a synthetic view of Th cell plasticity in terms of a graph connecting subtypes with arcs labeled by input conditions. Finally, we explore novel strategies enabling specific Th cell polarizing or reprograming events.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

7.1.1. RTOS Contract

Title: Static Analysis of a Fragment of an Operating-System with ASTRÉEA

Type: Service contract

Duration: June 2014 - December 2014

Partners: École Normale Supérieure (France), CNRS (France), Airbus France (France)

Inria contact: Antoine Miné

Abstract: The aim of the contract is to study the formal verification of the safety of a fragment of a small real-time multi-task operating system. The verification will be performed using the ASTRÉEA analyzer, by adapting and extending the model of parallel executions developed at École Normale Supérieure.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. AnaStaSec

Title: Static Analysis for Security Properties

Type: ANR générique 2014

Defi: Société de l'information et de la communication

Instrument: ANR grant

Duration: January 2015 - December 2018

Coordinator: Inria Paris-Rocquencourt (France)

Others partners: Airbus France (France), AMOSSYS (France), CEA LIST (France), Inria Rennes-Bretagne Atlantique (France), TrustInSoft (France)

Inria contact: Jérôme Feret

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

Abstract: An emerging structure in our information processing-based society is the notion of trusted complex systems interacting via heterogeneous networks with an open, mostly untrusted world. This view characterises a wide variety of systems ranging from the information system of a company to the connected components of a private house, all of which have to be connected with the outside.

It is in particular the case for some aircraft-embedded computer systems, which communicate with the ground through untrusted communication media. Besides, the increasing demand for new capabilities, such as enhanced on-board connectivity, e.g. using mobile devices, together with the need for cost reduction, leads to more integrated and interconnected systems. For instance, modern aircrafts embed a large number of computer systems, from safety-critical cockpit avionics to passenger entertainment. Some systems meet both safety and security requirements. Despite thorough segregation of subsystems and networks, some shared communication resources raise the concern of possible intrusions.

Some techniques have been developed and still need to be investigated to ensure security and confidentiality properties of such systems. Moreover, most of them are model-based techniques operating

only at architectural level and provide no guarantee on the actual implementations. However, most security incidents are due to attackers exploiting subtle implementation-level software vulnerabilities. Systems should therefore be analysed at software level as well (i.e. source or executable code), in order to provide formal assurance that security properties indeed hold for real systems.

Because of the size of such systems, and considering that they are evolving entities, the only economically viable alternative is to perform automatic analyses. Such analyses of security and confidentiality properties have never been achieved on large-scale systems where security properties interact with other software properties, and even the mapping between high-level models of the systems and the large software base implementing them has never been done and represents a great challenge. The goal of this project is to develop the new concepts and technologies necessary to meet such a challenge.

The project ANASTASECproject will allow for the formal verification of security properties of software-intensive embedded systems, using automatic static analysis techniques at different levels of representation: models, source and binary codes. Among expected outcomes of the project will be a set of prototype tools, able to deal with realistic large systems and the elaboration of industrial security evaluation processes, based on static analysis.

8.1.1.2. Verasco

Title: Formally-verified static analyzers and compilers

Type: ANR Ingénierie Numérique Sécurité 2011

Instrument: ANR grant

Duration: Septembre 2011 - September 2015

Coordinator: Inria (France)

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

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

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

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

8.1.1.3. AstréeA

Title: Static Analysis of Embedded Asynchronous Real-Time Software

Type: ANR Ingénierie Numérique Sécurité 2011

Instrument: ANR grant

Duration: January 2012 - December 2015

Coordinator: Airbus France (France)

Others partners: École normale supérieure (France)

Inria contact: Antoine Miné

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

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

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. MemCad

Type: IDEAS

Defi: Design Composite Memory Abstract Domains

Instrument: ERC Starting Grant

Objectif: Design Composite Memory Abstract Domains

Duration: October 2011 - September 2016

Coordinator: Inria (France)

Partner: None

Inria contact: Xavier Rival

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

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

8.2.1.2. MBAT

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

Type: Artemis Call 10

Instrument: FP7 project

Duration: November 2011 - October 2014

Coordinator: Daimler (Germany)

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

See also: https://artemis-ia.eu/project/29-mbat.html

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

8.3. International Initiatives

8.3.1. Participation In other International Programs

8.3.1.1. EXEK

Title: EXEcutable Knowledge

Type: DARPA

Instrument: DARPA Program

Program: Big Mechanism

Duration: July 2014 - December 2017

Coordinator: Harvard Medical School (Boston, USA)

Partner: Inria Paris-Rocquencourt, École normale supérieure de Lyon Université Paris-Diderot,

Inria contact: Jérôme Feret

Abstract: Our overarching objective is Executable Knowledge: to make modeling and knowledge representation twin sides of biological reasoning. This requires the definition of a formal language with a clear operational semantics for representing proteins and their interaction capabilities in terms of agents and rules informed by, but not exposing, biochemical and biophysical detail. Yet, to achieve Executable Knowledge we need to go further:

- Bridge the gap between rich data and their formal representation as executable model elements. Specifically, we seek an intermediate, but already formal, knowledge representation (meta-language) to express granular data germane to interaction mechanisms; a protocol defining which and how data are to be expressed in that language; and a translation procedure from it into the executable format.
- Implement mathematically sound, fast, and scalable tools for analyzing and executing arbitrary collections of rules.
- Develop a theory of causality and attendant tools to extract and analyze the unfolding of causal lineages to observations in model simulations.

We drive these technical goals with the biological objective of assembling rule-based models germane to Wnt signaling in order to understand the role of combinatorial complexity in robustness and control.

8.3.2. Inria International Labs

Xavier Rival attended the LIAMA Open Day in July 2014, gave a talk on “Modular Construction of Shape-Numeric Analyzers” and participated to the associated Summer School, giving a one day introduction to Verification by Abstract Interpretation.

8.3.3. Inria International Partners

8.3.3.1. Informal International Partners

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

Research on sensitivity is done in partnership with the group of Sukyoung Ryu (Assistant Professor at KAIST, Daejeon, Korea).

Research on numeric abstract domain is done in partnership with the groups of Ji Wang and Liqian Chen (National University of Defense Technology, Changsha, China) and of Deepak Kapur (University of New Mexico, USA).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

Kwangkeun Yi (Professor at Seoul National University, Seoul, Korea) visited the group during two weeks in June-July 2014. Sukyoung Ryu (Assistant Professor at KAIST, Daejeon, Korea) visited the group during four weeks in July-August 2014.

8.4.2. Internships

Benjamin Audry accomplished a under the supervision of Jérôme Feret (while he was a student at "Collège du Parc", Sucy en Brie, France).

Pretesh Agrawal accomplished a pre-doctoral internship undern the supervision of Jérôme Feret and Norman Ferns (while he was a fourth year undergraduate student at IIT Kanpur, India).

Émile Ferreux and Nessim Morsli accomplished under the supervision of Jérôme Feret (while they were L1 student of the FDV Bachelor program, Frontiers in Life Science, at University Paris-Descartes, France).

Huisong Li accomplished a pre-doctoral internship under the supervision of Xavier Rival (while she was a student at the Institute of Software, at the Chinese Academy of Sciences (Beijing, China).

Thibault Suzanne accomplished a Master internship under the supervision of Antoine Miné.

Abdelraouf Ouadjaout, a PhD student at CERIST Research Center (Alger), performed a one-month internship in the group under the supervision of Antoine Miné. The internship was funded by the Ministry of Higher Education and Scientific Research of Algeria.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. member of the steering program committee

Jérôme Feret is member of the Steering Committee of the international Workshop on Static Analysis and Systems Biology (SASB).

9.1.1.2. member of the organizing committee

Tie Cheng was a member of the Organizing Committtee of EuSpRIG 2014.

Antoine Miné was a member of the Organizing Committtee of the 5th Workshop on Tools for Automatic Program Analysis (TAPAS 2014), Munich, Germany, 10 September, 2014.

9.1.2. Scientific events selection

9.1.2.1. responsable of the conference program committee

Xavier Rival was Co-Chair of the Program Committee of the 15th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2014), San Diego, USA, January 19-21, 2014, [32]).

9.1.2.2. member of the conference program committee

Jérôme Feret was member of the Program Committee of the 15th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2014), San Diego, USA, January 19–21, 2014; of the Program Committee of the Sixth International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2014), CHamonix, France, April 20–24, 2014; of the Program Committee of the 10th International Workshop on Developments in Computational Models (DCM 2014), Vienna, Austria, July 13 2014; of the Program Committee of the International Workshop on Verification of Engineered Molecular Devices and Programs (VEMDP 2014), Vienna, Austria, July 17, 2014; of the Program Committee of the 3rd International Conference on Biomedical Engineering and Biotechnology (iCBEB 2014), Beijing, China, August 18–20, 2014. of the Program Committee of the 8th IFIP International Conference on Theoretical Computer Science (TCS 2014), Rome, Italy, September 1–3, 2014; of the Program Committee of the 5th International Workshop on Static Analysis and Systems Biology (SASB 2014), München, Germany, September 10, 2014; of the Program Committee of the 19th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2014), Florence, Italy, September 11–12, 2014; Jérôme Feret is member of the Program Committee of of the Program Committee of 11th International Workshop on Developments in Computational Models (DCM 2015), Cali, Colombia, October 28, 2015; of the Program Committee of the 7th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies (BIOTECHNO 2015), Rome, Italy, May 24–29, 2015; of the Program Committee of the 24th International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR 2015), Siena, Italy, July 13–15, 2015.

Xavier Rival was member of the Program Committee of the 41st ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2014 PC), San Diego, USA, January 22-24, 2014; of the Program Committee of the 21st International Static Analysis Symposium (SAS 2014), München, Germany, September 11–13, 2014; of the Extended Review Committee of the 42nd ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2015 ERC), Mumbai, India, January 15-17, 2015.

Antoine Miné was member of the Program Committee of the 5th International Workshop on Polyhedral Compilation Techniques (IMPACT'5), Amsterdam, The Netherlands, the 11th International Conference on Integrated Formal Methods (iFM'14), Bertinoro, Italy, 9–11 September 2014, the 11th School on MOdelling and VErifying parallel Processes (MOVEP'14), Nantes, France, 7–11 July 2014, and the 8th International Symposium on Theoretical Aspects of Software Engineering (TASE'14), Changsha, China, 1–3 September 2014. Antoine Miné was member of the External Review Committee of the 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'14), Mumbai, India, 11-18 January 2015. Antoine Miné is member of the Program Committee of the 22nd International Static Analysis Symposium (SAS'15), Saint-Malo, France, 9–11 September 2015, and the 9th International Symposium on Theoretical Aspects of Software Engineering (TASE'15), Nanjing, China, 12–14 September 2015.

Caterina Urban is member of the Competition Jury for the 4th Competition on Software Verification (SV-COMP 2015) to be held at TACAS 2015 in London, UK.

9.1.2.3. reviewer

Arlen Cox was a reviewer for the 21st International Static Analysis Symposium (SAS 2014), München, Germany, September 11–13, 2014.

Jérôme Feret was a reviewer for the 42nd ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2015), Mumbai, India, January 15–17, 2015, and for the 18th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2015), London, UK, April 11–18, 2015.

Xavier Rival was a reviewer for the Symposium on Programming Languages Design and Implementation (PLDI 2014), Edimburgh, UK, June 9–11, 2014, and for the AVICPS Workshop, Rome, Italy, December 2, 2014.

Antoine Miné was a reviewer for the 12th Asian Symposium on Programming Languages and Systems (APLAS 2014), Singapore, November 17-19, 2014, the 35th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI 2015), Edinburgh, UK, 9–11 June 2014, and the 21st International Static Analysis Symposium (SAS 2014), Munich, Germany, September 11–13, 2014.

Caterina Urban was reviewer for the conference Theoretical Computer Science (TCS), Roma, Italy, 1–3 September 2014, and the 26th International Conference on Computer Aided Verification (CAV 2014), Vienna, Austria, 18–22 July, 2014.

9.1.3. Journal

9.1.3.1. member of the editorial board

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

Antoine Miné is member of the editorial board of The Scientific World Journal and of Conference Papers in Computer Science.

9.1.3.2. reviewer

In 2014, Jérôme Feret was a reviewer for BMC Systems Biology; Software, Practice and Experience; Transactions on Modeling and Computer Simulation; PLOS Computational Biology; BMC Bioinformatics; and Axioms.

Antoine Miné was a reviewr for the Computer Languages, Systems & Structures journal, the Software and Systems Modeling journal, and the Transactions on Programming Languages and Systems journal.

9.1.4. Grants selection

9.1.4.1. reviewer

Jérôme Feret was a reviewer for the Swiss National Science Foundation.

9.1.5. Participation in Conferences

CAV: Arlen Cox and Caterina Urban attended the conference on Computer Aided Verification (Vienna, Austria, July 22, 2014). Arlen Cox gave a talk on “QUICr: A Parametric Abstraction Domain for Sets of Numbers”.

ECCB: Wassim Abou-Jaoudé attended the European Conference on Computational Biology (Strasbourg, France, September, 2014) and gave an invited talk on [8] at the workshop on Logical Modelling and Analysis of Cellular Networks.

ESOP: Antoine Miné and Caterina Urban attended the 23rd European Symposium on Programming (Grenoble, France); Caterina Urban presented an article [27].

EuSpRIG: Tie Cheng attended the Conference of the European Spreadsheet Risk Interest Group (EuSpRIG, July 2014).

ISOLA: Xavier Rival attended the International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (Corfu, Greece, October 2014), and presented an article [24].

ITP: Arnaud Spiwack attended the International Conference on Interactive Theorem Proving (ITP 2014, Vienna, July 2014), and presented an article [23].

JRSIF: Mehdi Bouaziz attended the research day of SIF “De la formation à l’innovation en Informatique”, (Paris, France, September 2014).

JSTools: Arlen Cox attended the workshop on JavaScript Tools (Uppsala, Sweden, July 28, 2014) and gave a talk on “JSAna with HOO: Automatically Verifying JavaScript Libraries without Client Code”.

LICS: Patrick Cousot, Jérôme Feret, and Caterina Urban attended the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014, Vienna, July 2014). Patrick Cousot gave an invited talk on [10].

LSB: Jérôme Feret attended the 5th Workshop on Logic and Systems Biology (LSB 2014, Vienna, July 2014) and gave an invited talk on [16].

NSAD: Mehdi Bouaziz, Tie Cheng, Arlen Cox, Antoine Miné, Xavier Rival, Antoine Toubhans and Caterina Urban attended the Workshop on Numerical and Symbolic Abstract Domains (NSAD 2014, München, Germany, September 2014).

OCaml meetup: Mehdi Bouaziz attended the OCaml meetups (Paris, France, September and December 2014). Thibault Suzanne attended an OCaml meetup and gave a talk on "Implémentation d'une structure de données : Hash Array Mapped Tries".

POPL: Mehdi Bouaziz, Patrick Cousot, Radhia Cousot, Arlen Cox, Jérôme Feret, Antoine Miné, Xavier Rival and Caterina Urban attended the Symposium on Principles Of Programming Languages (POPL 2014, San Diego, USA); Xavier Rival chaired a session; Caterina Urban presented a poster.

SAS: Mehdi Bouaziz, Tie Cheng, Patrick Cousot, Arlen Cox, Jérôme Feret, Antoine Miné, Xavier Rival, Antoine Toubhans, Caterina Urban, and Thibault Suzanne attended the Static Anaysis Symposium (SAS 2014, München, Germany, September 2014); Xavier Rival chaired a session; Arlen Cox presented an article [14]; Antoine Toubhans presented an article [25]; Caterina Urban presented an article [26].

SASB: Wassim Abou-Jaoudé and Jérôme Feret attended the Workshop on Static Analysis and Systems Biology (SASB 2014, München, Germany, September 2014). Jérôme Feret presented an article [16].

TAPAS: Mehdi Bouaziz, Tie Cheng, Patrick Cousot, Arlen Cox, Antoine Miné, Xavier Rival, Antoine Toubhans and Caterina Urban attended the Workshop on Tools for Automatic Program AnalysiS (TAPAS 2014, München, Germany, September 2014).

TYPES: Mehdi Bouaziz attented the Types 2014 meeting (TYPES 2014), Paris, France, May 2014).

VEMDP: Jérôme Feret attended the International Workshop on Verification of Engineered Molecular Devices and Programs (VEMDP 2014, Vienna, Austria, July 2014).

VMCAI: Mehdi Bouaziz, Patrick Cousot, Radhia Cousot, Arlen Cox, Jérôme Feret, Antoine Miné, Xavier Rival and Caterina Urban attended the Conference on Verification, Model-Checking and Abstract Interpretation (VMCAI 2014, San Diego, USA); Xavier Rival chaired the conference [32]; Jérôme Feret chaired a session; Antoine Miné presented an article [21].

WODES: Wassim Abou-Jaoudé attended the International Workshop on Discrete Event Systems (WODES 2014, Cachan, France, May 2014).

WST: Caterina Urban attended the 14th International Workshop on Termination and presented an article [30].

9.1.6. Seminars and talks

- Wassim Abou-Jaoudé gave some talks on "Logical modeling of T-helper cell differentiation and plasticity" [8] at Instituto Gulbenkian de Ciência (Oeiras, Portugal) in June 2014 and Max Planck Institute for Molecular Genetics (Berlin, Germany) in November 2014.
- Mehdi Bouaziz attended to the Dagstuhl Seminar on Next Generation Static Analysis Tools (Wadern, Germany) in August 2014 and gave two talks on "Introduction to Abstract Interpretation: static analysis and beyond" at the séminaire des doctorants du département d'informatique de l'École normale supérieure (ENS Paris, Paris, France) in November 2014, and at the Séminaire Francilien de Sûreté de Fonctionnement, (École Centrale Paris) in November 2014.

- Arlen Cox gave a talk on "HOO: Heap with Open Objects" at the Dagstuhl Seminar for Scripting Languages and Frameworks: Analysis and Verification (Wadem, Germany) in June 2014, a talk on "Statically Analyzing Object-Manipulating JavaScipt Libraries without Client Code" at the Institute for Defense Analysis (Bowie, Maryland, USA) in September 2014, and talks on "JSAna with HOO: Automatically Verifying JavaScript Libraries without Client Code" at Facebook (London, U.K) in August 2014, at Cyberpoint Lab (Baltimore, Maryland, USA) in September 2014, at Apple (Cupertino, California, USA) in November 2014, at Draper Labs (Boston, Massachussets, USA) in November 2014, and at MathWorks (Natick, Massachussets, USA) in November 2014.
- Jérôme Feret gave a talk "An algebraic approach for inferring and using symmetries in rule-based models" at the Dagstuhl Seminar on Next Generation Static Analysis Tools (Wadern, Germany) in August 2014 and a talk on "Rule-Based Modeling" at the Dagstuhl Seminar on Multiscale Spatial Computational Systems Biology (Wadern, Germany) in November 2014
- Xavier Rival gave a talk "Towards precise and scalable static analyses" at the 54th IFIP WG 2.4 Group Meeting (Monterey, California, USA) in February 2014, a talk on "Verification of safety critical embedded software" at the Retraite du Département d'Informatique (Belleme, Normandie, France) in May 2014, talks on "Modular Construction of Shape-Numeric Analyzers" at Aarhus University (Aarhus, Denmark) in June 2014, at East China Normal University (Shanghai, China) in July 2014 and at Northern University for Defense Technology (Changsha, China) in July 2014, a talk on "Construction of abstract domains for heterogeneous memory properties" at the Dagstuhl Seminar on Next Generation Static Analysis Tools (Wadern, Germany) in August 2014, and a talk on "Type Verification of Spreadsheet Applications" at the 55th IFIP WG 2.4 Group Meeting (Stellenbosch, South Africa) in November 2014.
- Antoine Toubhans gave a talk on "Composite abstract domains for shape analysis" at the Seminar of the Gallium Group (Inria Paris-Rocquencourt), in September 2014.
- Antoine Miné gave a talk on " A method to infer inductive numeric invariants inspired from Constraint Programming" at Decision Procedures and Abstract Interpretation (Dagstugh Seminar 14351) in August 2014, a seminar on "Mixing abstract interpretation and constraint programming techniques" at Laboratoire VERIMAG (Grenoble, France) in October 2014, a poster presentation on "The Astrée static analyzer" at Evaluating Software Verification Systems: Benchmarks and Competitions (Dagstugh Seminar 14171) in April 2014.
- Caterina Urban gave a seminar on "Proving Guarantee and Recurrence Temporal Properties by Abstract Interpretation" at University of Udine (Italy) in December 2014, a seminar on "Proving Guarantee and Recurrence Temporal Properties by Abstract Interpretation" at ETH Zurich in November 2014, a seminar "Automatic Inference of Ranking Functions by Abstract Interpretation" at Queen Mary University of London in October 2014, a talk on "Automatic Inference of Ranking Functions by Abstract Interpretation" at Dagstuhl Seminar 14352 "Next Generation Static Software Analysis Tools" in August 2014, a seminar on "Automatic Inference of Ranking Functions by Abstract Interpretation" at University College London in June 2014, a seminar on "An Abstract Domain to Infer Ordinal-Valued Ranking Functions" at Inria Rennes, Rennes in May 2014, a seminar on "Automatic Inference of Ranking Functions by Abstract Interpretation" at Séminaire du DI ENS in May 2014, a seminar on "Automatic Inference of Ranking Functions by Abstract Interpretation" at Inria Paris-Rocquencourt in March 2014, a seminar on "Automatic Inference of Ranking Functions by Abstract Interpretation" at IBM Thomas J. Watson Research Center in January 2014.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Licence:
  - Mehdi Bouaziz, Concepts Informatique (CI2), 22 ETD, L1, Université Paris-Diderot, Paris, France.

  - Mehdi Bouaziz, Intensive training in algorithmics, 35 ETD, Epita, Le Kremlin-Bicètre, octobre-novembre 2014.
  - Mehdi Bouaziz, Preparation to the International Olympiad in Informatics, 40h ETD, Epita, Le Kremlin-Bicètre, France.
  - Tie Cheng, Introduction to algorithmics, 36h ETD, L3, École Polytechnique, Palaiseau, France.
  - Jérôme Feret, and Caterina Urban, Mathematics, 40h ETD, L1, FDV Bachelor program (Frontiers in Life Sciences (FdV)), Université Paris-Descartes, France.
  - Antoine Miné and Xavier Rival, Semantics and applications to verification, 48h ETD, L3, École Normale Supérieure, Paris, France.
  - Xavier Rival, Introduction to static analysis, 8h, L3, École des Mines de Paris, Paris, France.
  - Xavier Rival, Introduction to algorithmics, 40h ETD, L3, École Polytechnique, Palaiseau, France.
  - Antoine Toubhans, Functional Analysis Class, 34h ETD, L2, Université Pierre et Marie Curie (UPMC), Paris, France.
  - Caterina Urban, Mathematics, 26.5h ETD, Bachelor Program "Frontières du Vivant", Université Paris Descartes.
  - Thibault Suzanne, Introduction to Programming, 72h ETD, L1, Université Paris Diderot.
- Master:
  - Vincent Danos and Jérôme Feret (with Jean Krivine), Computational Biology, 20h ETD, M1. Interdisciplinary Approaches to Life Science (AIV), Master Program, Université Paris-Descartes, France.
  - Jérôme Feret, Antoine Miné, and Xavier Rival, Abstract Interpretation: application to verification and static analysis, 72h ETD, M2. Parisian Master of Research in Computer Science (MPRI). École normale supérieure. France.
  - Xavier Rival, Introduction to abstract interpretation, 8h, M1-M2, East China Normal University, Shanghai, China.

9.2.2. Supervision

PhD:

- Arlen Cox, Parametric heap abstraction for dynamic languages libraries, University of Colorado at Boulder (USA), 17 of November 2014, supervised jointly by Bor-Yuh Evan Chang (University of Colorado), Xavier Rival (École Normale Supérieure) and Sriram Sankaranarayana (University of Colorado).

PhD in progress:

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

- Vincent Laviron, Static Analysis of Functional Programs by Abstract Interpretation, October 2009, Patrick Cousot, École Normale Supérieure.
- Huisong Li, Static Analysis of Data-Structures with Sharing, November 2014, Xavier Rival, École Normale Supérieure
- Jiangchao Liu, Verification of the memory safety of a micro-kernel, December 2013, Xavier Rival, École Normale Supérieure
- Antoine Toubhans, Combination of shape abstract domains, October 2011, Xavier Rival, École Doctorale de Paris Centre
- Caterina Urban, Static Analysis of Functional Temporal Properties of Programs by Abstract Interpretation, November 2011, Radhia Cousot and Antoine Miné, École Normale Supérieure.
- Thibault Suzanne, Thread-modular static analysis of concurrent programs in the presence of weak memory models, October 2014, Antoine Miné, École Normale Supérieure.

9.2.3. Juries

Jérôme Feret was in the jury of the PhD thesis of Arlen Cox (17 of November 2014, University of Colorado, Boulder, USA). Jérôme Feret was reviewer of the PhD thesis of Peter Kreyßig (to be defended, Friedrich Schiller University, Jena, Germany).

Xavier Rival took part to the jury of the Habilitation Thesis of Francesco Zappa-Nardelli (16th of January 2014, ENS Paris) and to the jury of the Habilitation Thesis of Bertrand Jeannet (27 of March 2014, Université Joseph Fourrier, Grenoble).

Antoine Miné is reviewer for the PhD thesis of Vivien Maisonneuve (École des Mines de Paris, February 2015). Antoine Miné was reviewer and in the jury of the PhD thesis of Julien Henry (Université de Grenoble, 13 October 2014).

9.3. Popularization

Mehdi Bouaziz gave a course on “Apprendre et enseignement — la programmation informatique des bibliothèques”, Maire de Paris, 12 hours, june-july 2014. Xavier Rival gave a talk on “Safety critical embedded softwares and their verification” at the Ceremony for the “Médailles of Engeniering Sciences” in April 2014. Xavier Rival participated to the organization of the “Nuit des Sciences” in June 2014.

10. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

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

Publications of the year

Articles in International Peer-Reviewed Journals

[8] W. ABOU-JAOUDÉ, P. T. MONTEIRO, A. NALDI, M. GRANDCLAUDON, V. SOUMELIS, C. CHAOUIYA, D. THIEFFRY. *Model checking to assess T-helper cell plasticity*, in "Frontiers in Bioengineering and Biotechnology", 2014, 22 p. [*DOI :* 10.3389/FBIOE.2014.00086], https://hal.inria.fr/hal-01099490

[9] J. FERET, H. KOEPPL, T. PETROV. *Stochastic Fragments*, in "International Journal of Software and Informatics", 2014, vol. 7, n^{o} 4, pp. 527-604, https://hal.inria.fr/hal-01098561

Invited Conferences

[10] P. COUSOT, R. COUSOT. *Abstract interpretation: past, present and future*, in "Symposium on Logic in Computer Science (LICS)", Vienna, Austria, July 2014 [*DOI :* 10.1145/2603088.2603165], https://hal.archives-ouvertes.fr/hal-01108790

[11] A. MINÉ. *AstréeA: A Static Analyzer for Large Embedded Multi-Task Software*, in "16th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'15)", Mumbai, India, Lecture Notes in Computer Science, Springer, January 2015, vol. 8931, 3 p. , https://hal.inria.fr/hal-01105235

International Conferences with Proceedings

[12] L. CHEN, J. LIU, A. MINÉ, D. KAPUR, J. WANG. *An Abstract Domain to Infer Octagonal Constraints with Absolute Value*, in "21st International Static Analysis Symposium (SAS'14)", Munich, Germany, Lecture Note in Computer Science, Springer, September 2014, vol. 8373, 18 p. [*DOI :* 10.1007/978-3-319-10936-7_7], https://hal.inria.fr/hal-01105217

[13] T. CHENG, X. RIVAL. *Static Analysis for Spreadsheet Applications for Type-Unsafe Operations Detection*, in "European Symposium On Programming (ESOP 2015)", London, United Kingdom, Proceedings of the European Symposium On Programming, April 2015, https://hal.archives-ouvertes.fr/hal-01098377

[14] A. COX, B.-Y. E. CHANG, X. RIVAL. *Automatic Analysis of Open Objects in Dynamic Language Programs*, in "21st Static Analysis Symposium (SAS)", München, Germany, 21st Static Analysis Symposium Proceedings, September 2014, n^{o} 8723 [*DOI :* 10.1007/978-3-319-10936-7_9], https://hal.archives-ouvertes.fr/hal-01095955

[15] A. COX, B.-Y. E. CHANG, X. RIVAL. *Desynchronized Multi-State Abstractions for Open Programs in Dynamic Languages*, in "European Symposium On Programming (ESOP 2015)", London, United Kingdom, Proceedings of the European Symposium On Programming (ESOP 2015), Springer, April 2015, https://hal.archives-ouvertes.fr/hal-01098379

[16] J. FERET. *An algebraic approach for inferring and using symmetries in rule-based models*, in "Static Analysis and Systems Biology", Munich, Germany, H. KOEPPL, L. PAULEVÉ (editors), Electronic Notes in Theoretical Computer Science, Elsevier, September 2014, 20 p. , forthcoming, https://hal.inria.fr/hal-01098556

[17] N. FERNS, S. KNIGHT, D. PRECUP. *Bisimulation for Markov Decision Processes through Families of Functional Expressions*, in "Horizons of the Mind. A Tribute to Prakash Panangaden (for his 60th birthday)", Oxford, United Kingdom, F. VAN BREUGEL, E. KASHEFI, C. PALAMIDESSI, J. RUTTEN (editors), Horizons of the Mind. A Tribute to Prakash Panangaden, Springer, May 2014, vol. 8464, pp. 319-342 [*DOI :* 10.1007/978-3-319-06880-0_17], https://hal.inria.fr/hal-01098566

[18] N. FERNS, D. PRECUP. *Bisimulation Metrics are Optimal Value Functions*, in "The 30th Conference on Uncertainty in Artificial Intelligence", Quebec City, Canada, N. L. ZHANG, J. TIAN (editors), AUAI Press, July 2014, 10 p. , https://hal.inria.fr/hal-01101180

[19] H. HERBELIN, A. SPIWACK. *The Rooster and the Syntactic Bracket* , in "19th International Conference on Types for Proofs and Programs (TYPES 2013)", Toulouse, France, Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, July 2014, vol. 26, pp. 169–187 [*DOI :* 10.4230/LIPICS.TYPES.2013.169], https://hal.inria.fr/hal-01097919

[20] J. LIU, X. RIVAL. *Abstraction of Arrays Based on Non Contiguous Partitions*, in "16th International Conference on Verification, Model Checking, and Abstract Interpretation", Mumbai, India, Proceedings of the 16th Conference on Verification, Model Checking, and Abstract Interpretation, Springer, January 2015, n^{o} 8931, pp. 282 - 299 [*DOI :* 10.1007/978-3-662-46081-8_16], https://hal.archives-ouvertes.fr/hal-01095985

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

[22] P. T. MONTEIRO, W. ABOU-JAOUDÉ, D. THIEFFRY, C. CHAOUIYA. *Model Checking logical regulatory networks*, in "WODES'14, 12th IFAC - IEEE International Workshop on Discrete Event Systems", Cachan, France, J.-J. LESAGE, J. E. RIBEIRO, B. LENNARTSON (editors), Elsevier, May 2014 [*DOI :* 10.3182/20140514-3-FR-4046.00135], https://hal.inria.fr/hal-01099489

[23] G. NAVES, A. SPIWACK. *Balancing Lists: A Proof Pearl*, in "ITP - 5th International Conference on Interactive Theorem Proving", Vienna, Austria, Interactive Theorem Proving, July 2014, vol. 8558, pp. 437 - 449 [*DOI :* 10.1007/978-3-319-08970-6_28], https://hal.inria.fr/hal-01097937

[24] X. RIVAL, A. TOUBHANS, B.-Y. E. CHANG. *Construction of Abstract Domains for Heterogeneous Properties (Position Paper)*, in "Leveraging Applications of Formal Methods, Verification and Validation", Corfu, Greece, Leveraging Applications of Formal Methods, Verification and Validation (ISOLA), Specialized Techniques and Applications, Springer, October 2014, n^{o} 8803, pp. 489 - 492 [*DOI :* 10.1007/978-3-662-45231-8_40], https://hal.archives-ouvertes.fr/hal-01095977

[25] A. TOUBHANS, B.-Y. E. CHANG, X. RIVAL. *An Abstract Domain Combinator for Separately Conjoining Memory Abstractions*, in "Static Analysis Symposium", München, Germany, Springer, September 2014, n° 8723 [*DOI :* 10.1007/978-3-319-10936-7_18], https://hal.archives-ouvertes.fr/hal-01095934

[26] C. URBAN, A. MINÉ. *A Decision Tree Abstract Domain for Proving Conditional Termination*, in "21st International Static Analysis Symposium (SAS'14)", Munich, Germany, Lecture Notes in Computer Science, Springer, September 2014, vol. 8373, 17 p. [*DOI :* 10.1007/978-3-319-10936-7_19], https://hal.inria.fr/hal-01105221

[27] C. URBAN, A. MINÉ. *An abstract domain to infer ordinal-valued ranking functions*, in "ESOP 2014 - 23rd European Symposium on Programming", Grenoble, France, Z. SHAO (editor), Springer, April 2014, vol. 8410, pp. 412-431 [*DOI :* 10.1007/978-3-642-54833-8_22], https://hal.inria.fr/hal-00925731

[28] C. URBAN, A. MINÉ. *Proving Guarantee and Recurrence Temporal Properties by Abstract Interpretation*, in "16th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'15)", Mumbai, India, Lecture Notes in Computer Science, Springer, January 2015, vol. 8931, 19 p. [*DOI :* 10.1007/978-3-662-46081-8_11], https://hal.inria.fr/hal-01105238

Conferences without Proceedings

[29] A. MINÉ. *A method to infer inductive numeric invariants inspired from Constraint Programming*, in "Decision Procedures and Abstract Interpretation (Dagstugh Seminar 14351)", Warden, Germany, Dagstuhl Seminar, August 2014, n° 14351, https://hal.inria.fr/hal-01105401

[30] C. URBAN, A. MINÉ. *To Infinity... and Beyond!*, in "14th International Workshop on Termination (WST'14)", Vienne, Austria, July 2014, 5 p. , https://hal.inria.fr/hal-01105216

[31] C. URBAN. *FuncTion: An Abstract Domain Functor for Termination*, in "21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems", Londres, United Kingdom, Lecture Notes in Computer Science, Springer, April 2015, 3 p. , https://hal.inria.fr/hal-01107419

Scientific Books (or Scientific Book chapters)

[32] *Verification, Model Checking, and Abstract Interpretation - 15th International Conference, VMCAI 2014, San Diego, CA, USA, January 19-21, 2014, Proceedings*, SpringerFrance, January 2014, n° 8318 [*DOI :* 10.1007/978-3-642-54013-4], https://hal.inria.fr/hal-01095535

Other Publications

[33] T. SUZANNE. *Analyse statique par interprétation abstraite de programmes concurrents dans le modèle de mémoire faible TSO*, Paris 7, September 2014, 20 p. , https://hal.inria.fr/hal-01105251

References in notes

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

[35] 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 GrenobleGrenoble, France, 21 March 1978

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

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

[38] P. COUSOT. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation*, in "Electr. Notes Theor. Comput. Sci.", 1997, vol. 6, pp. 77–102, http://dx.doi.org/10.1016/S1571-0661(05)80168-9

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

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

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

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

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

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

[45] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. HAYMAN, J. KRIVINE, C. THOMPSON-WALSH, G. WINSKEL. *Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models*, in "32nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)", Hyderabad, India, D. D'SOUZA, J. RADHAKRISHNAN, K. TELIKEPALLI (editors), LIPIcs, December 2012, http://hal.inria.fr/hal-00734487

[46] V. DANOS, J. FERET, W. FONTANA, R. HARMER, J. KRIVINE. *Abstracting the differential semantics of rule-based models: exact and automated model reduction*, in "Proceedings of Logic in Computer Science (LICS 2010), Edinburgh, UK", J.-P. JOUANNAUD (editor), 2010, pp. 362–381, http://hal.inria.fr/hal-00520112

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

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

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

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

[51] J. DESHARNAIS, R. JAGADEESAN, V. GUPTA, P. PANANGADEN. *The Metric Analogue of Weak Bisimulation for Probabilistic Processes*, in "LICS", IEEE Computer Society, 2002, pp. 413-422

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

[53] R. HARMER, V. DANOS, J. FERET, J. KRIVINE, W. FONTANA. *Intrinsic Information carriers in combinatorial dynamical systems*, in "Chaos", 2010, vol. 20, n^{o} 3, 037108, http://hal.inria.fr/hal-00520128

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

[55] F. VAN BREUGEL, J. WORRELL. *An Algorithm for Quantitative Verification of Probabilistic Transition Systems*, in "CONCUR", K. G. LARSEN, M. NIELSEN (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2154, pp. 336-350

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