IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2018

Project-Team ANTIQUE

Static Analysis by Abstract Interpretation

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

RESEARCH CENTER
Paris

THEME
Proofs and Verification

Table of contents

Project-Team ANTIQUE

Creation of the Team: 2014 January 01, updated into Project-Team: 2015 April 01

Keywords:

Computer Science and Digital Science:

A2. - Software
A2.1. - Programming Languages
A2.1.1. - Semantics of programming languages
A2.1.7. - Distributed programming
A2.1.12. - Dynamic languages
A2.2.1. - Static analysis
A2.3. - Embedded and cyber-physical systems
A2.3.1. - Embedded systems
A2.3.2. - Cyber-physical systems
A2.3.3. - Real-time systems
A2.4. - Formal method for verification, reliability, certification
A2.4.1. - Analysis
A2.4.2. - Model-checking
A2.4.3. - Proofs
A2.6.1. - Operating systems
A4.4. - Security of equipment and software
A4.5. - Formal methods for security

Other Research Topics and Application Domains:

B1.1. - Biology
B1.1.8. - Mathematical biology
B1.1.10. - Systems and synthetic biology
B5.2. - Design and manufacturing
B5.2.1. - Road vehicles
B5.2.2. - Railway
B5.2.3. - Aviation
B5.2.4. - Aerospace
B6.1. - Software industry
B6.1.1. - Software engineering
B6.1.2. - Software evolution, maintenance
B6.6. - Embedded systems

1. Team, Visitors, External Collaborators

Research Scientists

Xavier Rival [Team leader, Inria, Senior Researcher, HDR]
Vincent Danos [CNRS, Senior Researcher, HDR]
Cezara Drăgoi [Inria, Researcher]

Jérôme Feret [Inria, Researcher]

PhD Students

Andreea Beica [Ecole Normale Supérieure Paris]
Gaëlle Candel [Keymetrics]
Marc Chevalier [Ecole Normale Supérieure Lyon]
Hugo Illous [Ecole Normale Supérieure Paris]
Huisong Li [Inria, until Feb 2018]
Thibault Suzanne [Ecole Normale Supérieure Paris]
Jiangchao Liu [Inria, until Mar 2018]

Technical staff

Ferdinanda Camporesi [Inria, until Apr 2018]
Yves Stan Le Cornec [Inria]

Interns

Andrei Nicolae Damian [Inria, from Jun 2018 until Sep 2018]
Aurelie Faure de Pebeyre [Inria, until Jun 2018]
Constantin Alexandru Militaru [Inria, from Jun 2018 until Sep 2018]
Amit Kiran Rege [Ecole Normale Supérieure Paris, from Feb 2018 until Jul 2018]
Albin Salazar [Inria, from Sep 2018]

Administrative Assistant

Nathalie Gaudechoux [Inria]

Visiting Scientist

Pierre Boutillier [Harvard Medical School]

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 become 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), medical systems (pacemakers, surgery and patient monitoring systems), and value transfers in decentralized systems (smart contracts), rely on complex software, which should be verified.

Beyond the field of embedded systems, other pieces of software may cause very significant harm in the 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 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 one to express 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* [27]. 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* [27], which over-approximate 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.

The same principles can be applied successfully to other goals. In particular, the abstract interpretation framework should be viewed as 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 [26], 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. From properties to explanations

In many application domains, we can go beyond the proof that a program satisfies its specification. Abstractions can also offer new perspectives to understand how complex behaviors of programs emerge from simpler computation steps. Abstractions can be used to find compact and readable representations of sets of traces, causal relations, and even proofs. For instance, abstractions may decipher how the collective behaviors of agents emerge from the orchestration of their individual ones in distributed systems (such as consensus protocols, models of signaling pathways). Another application is the assistance for the diagnostic of alarms of a static analyzer.

Complex systems and software have often times intricate behaviors, leading to executions that are hard to understand for programmers and also difficult to reason about with static analyzers. Shared memory and distributed systems are notorious for being hard to reason about due to the interleaving of actions performed by different processes and the non-determinism of the network that might lose, corrupt, or duplicate messages. Reduction theorems, e.g., Lipton's theorem, have been proposed to facilitate reasoning about concurrency, typically transforming a system into one with a coarse-grained semantics that usually increases the atomic sections. We investigate reduction theorems for distributed systems and ways to compute the coarse-grained counter part of a system automatically. Compared with shared memory concurrency, automated methods to reason about distributed systems have been less investigated in the literature. We take a programming language approach based on high-level programming abstractions. We focus on partially-synchronous communication closed round-based models, introduced in the distributed algorithms community for its simpler proof arguments. The high-level language is compiled into a low-level (asynchronous) programming language. Conversely, systems defined under asynchronous programming paradigms are decompiled into the high-level programming abstractions. The correctness of the compilation/decompilation process is based on reduction theorems (in the spirit of Lipton and Elrad-Francez) that preserve safety and liveness properties.

In models of signaling pathways, collective behavior emerges from competition for common resources, separation of scales (time/concentration), non linear feedback loops, which are all consequences of mechanistic interactions between individual bio-molecules (e.g., proteins). While more and more details about mechanistic interactions are available in the literature, understanding the behavior of these models at the system level is far from easy. Causal analysis helps explaining how specific events of interest may occur. Model reduction techniques combine methods from different domains such as the analysis of information flow used in communication protocols, and tropicalization methods that comes from physics. The result is lower dimension systems that preserve the behavior of the initial system while focusing of the elements from which emerges the collective behavior of the system.

The abstraction of causal traces offer nice representation of scenarios that lead to expected or unexpected events. This is useful to understand the necessary steps in potential scenarios in signaling pathways; this is useful as well to understand the different steps of an intrusion in a protocol. Lastly, traces of computation of a static analyzer can themselves be abstracted, which provides assistance to classify true and false alarms. Abstracted traces are symbolic and compact representations of sets of counter-examples to the specification of a system which help one to either understand the origin of bugs, or to find that some information has been lost in the abstraction leading to false alarms.

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 the 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, ASTRÉE successfully 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, and monitoring typically relies on a *parallel* structure, where several threads are executed concurrently, 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. Models of mechanistic interactions between proteins

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 one 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 analyses based on abstract interpretation, and 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 in order to identify key elements in the dynamics of these models. The results of these analyses are sent to another tool, which is used to automatically simplify 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.

4.4. Consensus

Fault-tolerant distributed systems provide a dependable service on top of unreliable computers and networks. Famous examples are geo-replicated data-bases, distributed file systems, or blockchains. Fault-tolerant protocols replicate the system and ensure that all (unreliable) replicas are perceived from the outside as one single reliable machine. To give the illusion of a single reliable machine "consensus" protocols force replicas to agree on the "current state" before making this state visible to an outside observer. We are interested in (semi-)automatically proving the total correctness of consensus algorithms in the benign case (messages are lost or processes crash) or the Byzantine case (processes may lie about their current state). In order to do this, we first define new reduction theorems to simplify the behaviors of the system and, second, we introduce new static analysis methods to prove the total correctness of adequately simplified systems. We focus on static analysis based Satisfiability Modulo Theories (SMT) solvers which offers a good compromise between automation and expressiveness. Among our benchmarks are Paxos, PBFT (Practical Byzantine Fault-Tolerance), and blockchain algorithms (Red-Belly, Tendermint, Algorand). These are highly challenging benchmarks, with a lot of non-determinism coming from the interleaving semantics and from the adversarial environment in which correct processes execute, environment that can drop messages, corrupt them, etc. Moreover, these systems were originally designed for a few servers but today are deployed on networks with thousands of nodes. The "optimizations" for scalability can no longer be overlooked and must be considered as integral part of the algorithms, potentially leading to specifications weaker than the so much desired consensus.

4.5. Models of growth

In systems and synthetic biology (engineered systems) one would like study the environment of a given cellular process (such as signaling pathways mentioned earlier) and the ways in which that process interacts with different resources provided by the host. To do this, we have built coarse-grained models of cellular physiology which summarize fundamental processes (transcription, translation, transport, metabolism). such models describe global growth in mechanistic way and allow one to plug the model of one's process of interest into a simplified and yet realistic and reactive model of the process interaction with its immediate environment. A first ODE-based deterministic version of this model [30] explaining the famous bacterial growth laws and how the allocation of resources to different genomic sectors depends on the growth conditions- was published in 2015 and has already received nearly 150 citations. The model also allows one to bridge between population genetic models which describe cells in terms of abstract features and fitness and intra-cellular models. For instance, we find that fastest growing strategies are not evolutionary stable in competitive experiments. We also find that vastly different energy storage strategies exist[16]. In a recent article[17] in *Nature Communications* we build a stochastic version of the above model. We predict the empirical size and doubling time distributions as a function of growth conditions. To be able to fit the parameters of the model to available single-cell data (note that the fitting constraints are far tighter than in the deterministic case), we introduce new techniques for the approximation of reaction-division systems which generalize continuous approximations of Langevin type commonly used for pure reaction systems. We also use cross-correlations to visualize causality and modes in noise propagation in the model (in a way reminiscent to abstract computational traces mentioned earlier). In other work, we show how to connect our new class of models to more traditional ones stemming from "flux balance analysis" by introducing an allocation vector which allows one to assign a formal growth rate to a class of reaction systems [25].

5. New Software and Platforms

5.1. APRON

SCIENTIFIC DESCRIPTION: The APRON library is intended to be a common interface to various underlying libraries/abstract domains and to provide additional services that can be implemented independently from the underlying library/abstract domain, as shown by the poster on the right (presented at the SAS 2007 conference. You may also look at:
FUNCTIONAL DESCRIPTION: 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.

- Participants: Antoine Miné and Bertrand Jeannet
- Contact: Antoine Miné
- URL: http://apron.cri.ensmp.fr/library/

5.2. Astrée

The AstréeA Static Analyzer of Asynchronous Software
KEYWORDS: Static analysis - Static program analysis - Program verification - Software Verification - Abstraction
SCIENTIFIC DESCRIPTION: 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.
FUNCTIONAL DESCRIPTION: 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.

Astrée is a static analyzer for sequential programs based on abstract interpretation. The Astrée static analyzer aims at proving the absence of runtime errors in programs written in the C programming language.

- Participants: Antoine Miné, Jérôme Feret, Laurent Mauborgne, Patrick Cousot, Radhia Cousot and Xavier Rival
- Partners: CNRS - ENS Paris - AbsInt Angewandte Informatik GmbH
- Contact: Patrick Cousot
- URL: http://www.astree.ens.fr/

5.3. AstréeA

The AstréeA Static Analyzer of Asynchronous Software
KEYWORDS: Static analysis - Static program analysis
SCIENTIFIC DESCRIPTION: 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.
FUNCTIONAL DESCRIPTION: AstréeA is a static analyzer prototype for parallel software based on abstract interpretation. The AstréeA prototype is a fork of the Astrée static analyzer that adds support for analyzing parallel embedded C software.

- Participants: Antoine Miné, Jérôme Feret, Patrick Cousot, Radhia Cousot and Xavier Rival
- Partners: CNRS - ENS Paris - AbsInt Angewandte Informatik GmbH
- Contact: Patrick Cousot
- URL: http://www.astreea.ens.fr/

5.4. ClangML

KEYWORD: Compilation

FUNCTIONAL DESCRIPTION: 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.

- Participants: Devin Mccoughlin, François Berenger and Pippijn Van Steenhoven
- Contact: Xavier Rival
- URL: https://github.com/Antique-team/clangml/tree/master/clang

5.5. FuncTion

SCIENTIFIC DESCRIPTION: 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. 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).
FUNCTIONAL DESCRIPTION: 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.

- Participants: Antoine Miné and Caterina Urban
- Contact: Caterina Urban
- URL: http://www.di.ens.fr/~urban/FuncTion.html

5.6. HOO

Heap Abstraction for Open Objects
FUNCTIONAL DESCRIPTION: 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.

- Participant: Arlen Cox
- Contact: Arlen Cox

5.7. MemCAD

The MemCAD static analyzer
KEYWORDS: Static analysis - Abstraction

FUNCTIONAL DESCRIPTION: MemCAD is 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.

- Participants: Antoine Toubhans, François Berenger, Huisong Li and Xavier Rival
- Contact: Xavier Rival
- URL: http://www.di.ens.fr/~rival/memcad.html

5.8. KAPPA

A rule-based language for modeling interaction networks
KEYWORDS: Systems Biology - Modeling - Static analysis - Simulation - Model reduction
SCIENTIFIC DESCRIPTION: OpenKappa is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language, a static analyzer (for debugging models), a simulator, a compression tool for causal traces, and a model reduction tool.
FUNCTIONAL DESCRIPTION: Kappa is provided with the following tools: - a compiler - a stochastic simulator - a static analyzer - a trace compression algorithm - an ODE generator.

RELEASE FUNCTIONAL DESCRIPTION: On line UI, Simulation is based on a new data-structure (see ESOP 2017), New abstract domains are available in the static analyzer (see SASB 2016), Local traces (see TCBB 2018), Reasoning on polymers (see SASB 2018).

- Participants: Jean Krivine, Jérôme Feret, Kim Quyen Ly, Pierre Boutillier, Russ Harmer, Vincent Danos and Walter Fontana
- Partners: ENS Lyon - Université Paris-Diderot - HARVARD Medical School
- Contact: Jérôme Feret
- URL: http://www.kappalanguage.org/

5.9. QUICr

FUNCTIONAL DESCRIPTION: 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.

- Participant: Arlen Cox
- Contact: Arlen Cox

5.10. LCertify

KEYWORD: Compilation
SCIENTIFIC DESCRIPTION: 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.

FUNCTIONAL DESCRIPTION: 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 semantic equivalence. 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.ntees that no compiler bug did cause incorrect code to be generated.

- Participant: Xavier Rival
- Partners: CNRS - ENS Paris
- Contact: Xavier Rival
- URL: http://www.di.ens.fr/~rival/lcertify.html

5.11. Zarith

FUNCTIONAL DESCRIPTION: 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 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.

- Participants: Antoine Miné, Pascal Cuoq and Xavier Leroy
- Contact: Antoine Miné
- URL: http://forge.ocamlcore.org/projects/zarith

6. New Results

6.1. A Theoretical Foundation of Sensitivity in an Abstract Interpretation Framework

Participants: Xavier Rival [correspondant], Sukyoung Ryu, Se-Won Kim.

In [14], we formalize a framework to design static analyses that make use of sensitivity, using the general notion of cardinal power abstraction.

Program analyses often utilize various forms of *sensitivity* such as context sensitivity, call-site sensitivity, and object sensitivity. These techniques all allow for more precise program analyses, that are able to compute more precise program invariants, and to verify stronger properties. Despite the fact that sensitivity techniques are now part of the standard toolkit of static analyses designers and implementers, no comprehensive frameworks allow the description of all common forms of sensitivity. As a consequence, the soundness proofs of static analysis tools involving sensitivity often rely on *ad hoc* formalization, which are not always carried out in an abstract interpretation framework. Moreover, this also means that opportunities to identify similarities between analysis techniques to better improve abstractions or to tune static analysis tools can easily be missed.

In this work, we formalize a framework for the description of *sensitivity in static analysis*. Our framework is based on a powerful abstract domain construction, and utilizes reduced cardinal power to tie basic abstract predicates to the properties analyses are sensitive to. We formalize this abstraction, and the main abstract operations that are needed to turn it into a generic abstract domain construction. We demonstrate that our approach can allow for a more precise description of program states, and that it can also describe a large set of sensitivity techniques, both when sensitivity criteria are static (known before the analysis) or dynamic (inferred as part of the analysis), and sensitive analysis tuning parameters. Last, we show that sensitivity techniques used in state of the art static analysis tools can be described in our framework.

6.2. Memory Abstraction

6.2.1. Abstraction of arrays based on non contiguous partitions

Participants: Jiangchao Liu, Xavier Rival [correspondant].

In [15], we studied the verification of components of embedded programs that utilize arrays to store dynamically chained data-structures. Furthermore, this work constitutes a significant part of Jiangchao Liu's PhD Thesis ([10]).

User-space programs rely on memory allocation primitives when they need to construct dynamic structures such as lists or trees. However, low-level OS kernel services and embedded device drivers typically avoid resorting to an external memory allocator in such cases, and store structure elements in contiguous arrays instead. This programming pattern leads to very complex code, based on data-structures that can be viewed and accessed either as arrays or as chained dynamic structures. The code correctness then depends on intricate invariants mixing both aspects. We propose a static analysis that is able to verify such programs. It relies on the combination of abstractions of the allocator array and of the dynamic structures built inside it. This approach allows to integrate program reasoning steps inherent in the array and in the chained structure into a single abstract interpretation. We report on the successful verification of several embedded OS kernel services and drivers.

6.2.2. Semantic-Directed Clumping of Disjunctive Abstract States

Participants: Huisong Li, Francois Berenger, Bor-Yuh Evan Chang, Xavier Rival [correspondant].

In [29], we studied the semantic directed clumping of disjunctive abstract states. Furthermore, this work constitutes a significant part of Huisong Li's PhD Thesis ([9]).

To infer complex structural invariants, Shape analyses rely on expressive families of logical properties. Many such analyses manipulate abstract memory states that consist of separating conjunctions of basic predicates describing atomic blocks or summaries. Moreover, they use finite disjunctions of abstract memory states in order to account for dissimilar shapes. Disjunctions should be kept small for the sake of scalability, though precision often requires to keep additional case splits. In this context, deciding when and how to merge case splits and to replace them with summaries is critical both for the precision and for the efficiency. Existing techniques use sets of syntactic rules, which are tedious to design and prone to failure. In this paper, we design a semantic criterion to clump abstract states based on their silhouette which applies not only to the conservative union of disjuncts, but also to the weakening of separating conjunction of memory predicates into inductive summaries. Our approach allows to define union and widening operators that aim at preserving the case splits that are required for the analysis to succeed. We implement this approach in the MemCAD analyzer, and evaluate it on real-world C codes from existing libraries, including programs dealing with doubly linked lists, red-black trees and AVL-trees.

6.3. Static Analysis of JavaScript Code

6.3.1. Weakly Sensitive Analysis for Unbounded Iteration over JavaScript Objects

Participants: Yoonseok Ko, Xavier Rival [correspondant], Sukyoung Ryu.

In [28], we studied composite object abstraction for the analysis JavaScript.

JavaScript framework libraries like jQuery are widely use, but complicate program analyses. Indeed, they encode clean high-level constructions such as class inheritance via dynamic object copies and transformations that are harder to reason about. One common pattern used in them consists of loops that copy or transform part or all of the fields of an object. Such loops are challenging to analyze precisely, due to weak updates and as unrolling techniques do not always apply. In this work, we observe that precise field correspondence relations are required for client analyses (e.g., for call-graph construction), and propose abstractions of objects and program executions that allow to reason separately about the effect of distinct iterations without resorting to full unrolling. We formalize and implement an analysis based on this technique. We assess the performance and precision on the computation of call-graph information on examples from jQuery tutorials.

6.4. Communication-closed asynchronous protocols

Participants: Andrei Damien, Cezara Drăgoi [correspondant], Alexandru Militaru, Josef Widder.

Fault-tolerant distributed systems are implemented over asynchronous networks, so that they use algorithms for asynchronous models with faults. Due to asynchronous communication and the occurrence of faults (e.g., process crashes or the network dropping messages) the implementations are hard to understand and analyze. In contrast, synchronous computation models simplify design and reasoning. In this paper, we bridge the gap between these two worlds. For a class of asynchronous protocols, we introduce a procedure that, given an asynchronous protocol, soundly computes its round-based synchronous counterpart. This class is defined by properties of the sequential code. We computed the synchronous counterpart of known consensus and leader election protocols, such as, Paxos, and Chandra and Toueg's consensus. Using Verifast we checked the sequential properties required by the rewriting. We verified the round-based synchronous counter-part of Multi-Paxos, and other algorithms, using existing deductive verification methods for synchronous protocols.

6.5. Borel Kernels and their Approximation, Categorically

Participants: Fredrik Dahlqvist, Alexandra Silva, Vicent Danos [correspondant], Ilias Garnier.

In [12] is introduced a categorical framework to study the exact and approximate semantics of probabilistic programs. We construct a dagger symmetric monoidal category of Borel kernels where the dagger-structure is given by Bayesian inversion. We show functorial bridges between this category and categories of Banach lattices which formalize the move from kernel-based semantics to predicate transformer (backward) or state transformer (forward) semantics. These bridges are related by natural transformations, and we show in particular that the Radon-Nikodym and Riesz representation theorems-two pillars of probability theory-define natural transformations. With the mathematical infrastructure in place, we present a generic and endogenous approach to approximating kernels on standard Borel spaces which exploits the involutive structure of our category of kernels. The approximation can be formulated in several equivalent ways by using the func-torial bridges and natural transformations described above. Finally, we show that for sensible discretization schemes, every Borel kernel can be approximated by kernels on finite spaces, and that these approximations converge for a natural choice of topology. We illustrate the theory by showing two examples of how approximation can effectively be used in practice: Bayesian inference and the Kleene * operation of ProbNetKAT.

6.6. Static analysis of rule-based models

Thanks to rule-based modeling languages, we can assemble large sets of mechanistic protein-protein interactions within integrated models. Our goal would be to understand how the behavior of these systems emerges from these low-level interactions. Yet this is a quite long term challenge and it is desirable to offer intermediary levels of abstraction, so as to get a better understanding of the models and to increase our confidence within our mechanistic assumptions. To this extend, static analysis can be used to derive various abstractions of the semantics, each of them offering new perspectives on the models.

6.6.1. Trace approximation

Participants: Jérôme Feret [correspondant], Kim Quyên Lý.

In [13], we propose an abstract interpretation of the behavior of each protein, in isolation. Given a model written in Kappa, this abstraction computes for each kind of proteins a transition system that describes which conformations this protein may take and how a protein may pass from one conformation to another one. Then, we use simplicial complexes to abstract away the interleaving order of the transformations between conformations that commute. As a result, we get a compact summary of the potential behavior of each protein of the model.

6.6.2. Detection of polymer formation

Participants: Pierre Boutillier, Aurélie Faure de Pebeyre, Jérôme Feret [correspondant].

Rule-based languages, such as Kappa and BNGL, allow for the description of very combinatorial models of interactions between proteins. A huge (when not infinite) number of different kinds of bio-molecular compounds may arise due to proteins with multiple binding and phosphorylation sites. Knowing beforehand whether a model may involve an infinite number of different kinds of bio-molecular compounds is crucial for the modeler. On the first hand, having an infinite number of kinds of bio-molecular compounds is sometimes a hint for modeling flaws: forgetting to specify the conflicts among binding rules is a common mistake. On the second hand, it impacts the choice of the semantics for the models (among stochastic, differential, hybrid).

In [22], we introduce a data-structure to abstract the potential unbounded polymers that may be formed in a rule-based model. This data-structure is a graph, the nodes and the edges of which are labeled with patterns. By construction, every potentially unbounded polymer is associated to at least one cycle in that graph. This data-structure has two main advantages. Firstly, as opposed to site-graphs, one can reason about cycles without enumerating them (by the means of Tarjan's algorithm for detecting strongly connected components). Secondly, this data-structures may be combined easily with information coming from additional reachability analysis: the edges that are labeled with an overlap that is proved unreachable in the model may be safely discarded.

6.6.3. The static analyzer KaSa

Participants: Pierre Boutillier, Ferdinanda Camporesi, Jean Coquet, Jérôme Feret [correspondant], Kim Quyên Lý, Nathalie Théret, Pierre Vignet.

KaSa is a static analyzer for Kappa models. Its goal is two-fold. Firstly, KaSa assists the modeler by warning about potential issues in the model. Secondly, KaSa may provide useful properties to check that what is implemented is what the modeler has in mind and to provide a quick overview of the model for the people who have not written it. The cornerstone of KaSa is a fix-point engine which detects some patterns that may never occur whatever the evolution of the system may be. From this, many useful information may be collected KaSa warns about rules that may never be applied, about potential irreversible transformations of proteins (that may not be reverted even thanks to an arbitrary number of computation steps) and about the potential formation of unbounded molecular compounds. Lastly, KaSa detects potential influences (activation/inhibition relation) between rules.

In [21], we illustrate the main features of KaSa on a model of the extracellular activation of the transforming growth factor, TGF-b.

6.7. The Kappa platform for rule-based modeling

Participants: Pierre Boutillier, Mutaamba Maasha, Xing Li, Héctor Medina-Abarca, Jean Krivine, Jérôme Feret [correspondant], Ioana Cristescu, Angus Forbes, Walter Fontana.

In [11], we present an overview of the Kappa platform, an integrated suite of analysis and visualization techniques for building and interactively exploring rule-based models. The main components of the platform are the Kappa Simulator, the Kappa Static Analyzer and the Kappa Story Extractor. In addition to these components, we describe the Kappa User Interface, which includes a range of interactive visualization tools for rule-based models needed to make sense of the complexity of biological systems. We argue that, in this approach, modeling is akin to programming and can likewise benefit from an integrated development environment. Our platform is a step in this direction.

We discuss details about the computation and rendering of static, dynamic, and causal views of a model, which include the contact map (CM), snapshots at different resolutions, the dynamic influence network (DIN) and causal compression. We provide use cases illustrating how these concepts generate insight. Specifically, we show how the CM and snapshots provide information about systems capable of polymerization, such as Wnt signaling. A well-understood model of the KaiABC oscillator, translated into Kappa from the literature, is deployed to demonstrate the DIN and its use in understanding systems dynamics. Finally, we discuss how pathways might be discovered or recovered from a rule-based model by means of causal compression, as exemplified for early events in EGF signaling.

The Kappa platform is available via the project website at kappa-language.org. All components of the platform are open source and freely available through the authors' code repositories.

6.8. Conservative approximation of systems of differential equations

We design a tools-kit to reason and abstract the solutions of the systems of differential equations that are described in high-level languages. Our abstractions are conservative in the sense that they provided sound lower and upper bounds for the value of some observables of the system. Our approach consists, firstly, in inferring structural equalities about combinations of variables and structural inequalities about the value of variable derivatives thanks to symbolic reasoning at the level of the languages and, then, in using these numerical constraints to infer two differential equations for the variables of interest — one for the lower bound and one for the upper bound.

We focus on the systems of equations that are described in Kappa. Our goal is to provide a unifying framework that can deal with heterogeneous kinds of abstractions, including truncation, time- and concentration-scale separations, flow-based reduction, symmetries-based reduction.

6.8.1. Approximation of models of polymers

Participants: Ken Chanseau Saint-Germain, Jérôme Feret [correspondant].

We propose a systematic approach to approximate the behavior of models of polymers synthesis/degradation, described in Kappa. Our abstraction consists in focusing on the behavior of all the patterns of size less than a given parameter. We infer symbolic equalities and inequalities which intentionally may be understood as algebraic constructions over patterns, and extensionally as sound properties about the concentration of the bio-molecular species that contain these patterns. Then, we derive a system of equations describing the time evolution of a lower and an upper bounds for the concentration of each pattern of interest.

This work has been presented at VEMDP 2018 (Verification of Engineered Molecular Devices and Programs), in Oxford, 19th July 2018, and at the days "BIOS-IA" of the working group BIOSS, at Pasteur Institute, Paris, 18th December 2018.

6.8.2. Approximation based on time- and/or concentration-scale separation

Participants: Andreea Beica, Jérôme Feret [correspondant].

In [20], we have designed and tested an approximation method for ODE models of biochemical reaction systems, in which the guarantees are our major requirement. Borrowing from tropical analysis techniques, we look at the dominance relations among terms of each species' ODE. These dominance relations can be exploited to simplify the original model, by neglecting the dominated terms. As the dominant subsystems can change during the system's dynamics, depending on which species dominate the others, several possible modes exist. Thus, simpler models consisting of only the dominant subsystems can be assembled into hybrid, piece-wise smooth models, which approximate the behavior of the initial system. By combining the detection of dominated terms with symbolic bounds propagation, we show how to approximate the original model by an assembly of simpler models, consisting in ordinary differential equations that provide time-dependent lower and upper bounds for the concentrations of the initial models species. The utility of our method is twofold. On the one hand, it provides a reduction heuristics that performs without any prior knowledge of the initial system's behavior (i.e., no simulation of the initial system is needed in order to reduce it). On the other hand, our method provides sound interval bounds for each species, and hence can serve to evaluate the faithfulness of tropicalization reduction heuristics for ODE models of biochemical reduction systems. The method is tested on several case studies.

6.9. Sources, propagation and consequences of stochasticity in cellular growth

Participants: Philipp Thomas, Guillaume Terradot, Vicent Danos [correspondant], Andrea Weiße.

Growth impacts a range of phenotypic responses. Identifying the sources of growth variation and their propagation across the cellular machinery can thus unravel mechanisms that underpin cell decisions.

In [17], we present a stochastic cell model linking gene expression, metabolism and replication to predict growth dynamics in single bacterial cells. Alongside we provide a theory to analyze stochastic chemical reactions coupled with cell divisions, enabling efficient parameter estimation, sensitivity analysis and hypothesis testing. The cell model recovers population-averaged data on growth-dependence of bacterial physiology and how growth variations in single cells change across conditions. We identify processes responsible for this variation and reconstruct the propagation of initial fluctuations to growth and other processes. Finally, we study drug-nutrient interactions and find that antibiotics can both enhance and suppress growth heterogeneity. Our results provide a predictive framework to integrate heterogeneous data and draw testable predictions with implications for antibiotic tolerance, evolutionary and synthetic biology.

6.10. Survival of the Fattest: Evolutionary Trade-offs in Cellular Resource Storage

Participants: Guillaume Terradot, Andreea Beica, Andrea Weiße, Vicent Danos [correspondant].

Cells derive resources from their environments and use them to fuel the bio-synthetic processes that determine cell growth. Depending on how responsive the bio-synthetic processes are to the availability of intracellular resources, cells can build up different levels of resource storage.

In [16], we use a recent mathematical model of the coarse-grained mechanisms that drive cellular growth to investigate the effects of cellular resource storage on growth. We show that, on the one hand, there is a cost associated with high levels of storage resulting from the loss of stored resources due to dilution. We further show that, on the other hand, high levels of storage can benefit cells in variable environments by increasing biomass production during transitions from one medium to another. Our results thus suggest that cells may face trade-offs in their maintenance of resource storage based on the frequency of environmental change.

6.11. A Genetic Circuit Compiler: Generating Combinatorial Genetic Circuits with Web Semantics and Inference

Participants: William Waites, Goksel Misirli, Matteo Cavaliere, Vicent Danos [correspondant].

A central strategy of synthetic biology is to understand the basic processes of living creatures through engineering organisms using the same building blocks. Biological machines described in terms of parts can be studied by computer simulation in any of several languages or robotically assembled in vitro. In [19] we present a language, the Genetic Circuit Description Language (GCDL) and a compiler, the Genetic Circuit Compiler (GCC). This language describes genetic circuits at a level of granularity appropriate both for automated assembly in the laboratory and deriving simulation code. The GCDL follows Semantic Web practice and the compiler makes novel use of the logical inference facilities that are therefore available. We present the GCDL and compiler structure as a study of a tool for generating κ-language simulations from semantic descriptions of genetic circuits.

6.12. An Information-Theoretic Measure for Patterning in Epithelial Tissues

Participants: William Waites, Matteo Cavaliere, Élise Cachat, Vicent Danos [correspondant], Jamie A. Davies.

In [18], we present path entropy, an information-theoretic measure that captures the notion of patterning due to phase separation in organic tissues. Recent work has demonstrated, both in silico and in vitro, that phase separation in epithelia can arise simply from the forces at play between cells with differing mechanical properties. These qualitative results give rise to numerous questions about how the degree of patterning relates to model parameters or underlying biophysical properties. Answering these questions requires a consistent and meaningful way of quantifying degree of patterning that we observe. We define a resolution-independent measure that is better suited than image-processing techniques for comparing cellular structures. We show how this measure can be usefully applied in a selection of scenarios from biological experiment and computer simulation, and argue for the establishment of a tissue-graph library to assist with parameter estimation for synthetic morphology.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. 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 analyzed 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 ANASTASEC project 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.

7.1.2. REPAS

The project REPAS, Reliable and Privacy-Aware Software Systems via Bisimulation Metrics (coordination Catuscia Palamidessi, Inria Saclay), aims at investigating quantitative notions and tools for proving program correctness and protecting privacy, focusing on bisimulation metrics, the natural extension of bisimulation on quantitative systems. A key application is to develop mechanisms to protect the privacy of users when their location traces are collected. Partners: Inria (Comete, Focus), ENS Cachan, ENS Lyon, University of Bologna.

7.1.3. SAFTA

Title: SAFTA Static Analysis for Fault-Tolerant distributed Algorithms.

Type: ANR JCJC 2018

Duration: February 2018 - February 2022

Coordinator: Cezara Drăgoi, CR Inria

Abstract: Fault-tolerant distributed data structures are at the core distributed systems. Due to the multiple sources of non-determinism, their development is challenging. The project aims to increase the confidence we have in distributed implementations of data structures. We think that the difficulty does not only come from the algorithms but from the way we think about distributed systems. In this project we investigate partially synchronous communication-closed round based programming abstractions that reduce the number of interleavings, simplifying the reasoning about distributed systems and their proof arguments. We use partial synchrony to define reduction theorems from asynchronous semantics to partially synchronous ones, enabling the transfer of proofs from the synchronous world to the asynchronous one. Moreover, we define a domain specific language, that allows the programmer to focus on the algorithm task, it compiles into efficient asynchronous code, and it is equipped with automated verification engines.

7.1.4. TGFSYSBIO

Title: Microenvironment and cancer: regulation of TGF-β signaling

Type: ANR générique 2014

Defi: Société de l'information et de la communication

Instrument: Plan Cancer 2014-2019

Duration: December 2015 - November 2018

Coordinator: INSERM U1085-IRSET

Others partners: Inria Paris (France), Inria Rennes-Bretagne Atlantique (France),

Inria contact: Jérôme Feret

Abstract: Most cases of hepatocellular carcinoma (HCC) develop in cirrhosis resulting from chronic liver diseases and the Transforming Growth Factor β (TGF-β) is widely regarded as both the major pro-fibrogenic agent and a critical inducer of tumor progression and invasion. Targeting the deleterious effects of TGF-β without affecting its physiological role is the common goal of therapeutic strategies. However, identification of specific targets remains challenging because of the pleiotropic effects of TGF-β linked to the complex nature of its extracellular activation and signaling networks.

Our project proposes a systemic approach aiming at to identifying the potential targets that regulate the shift from anti- to pro-oncogenic effects of TGF-β. To that purpose, we will combine a rule-based model (Kappa language) to describe extracellular TGF-beta activation and large-scale state-transition based (Cadbiom formalism) model for TGF-β-dependent intracellular signaling pathways. The multi-scale integrated model will be enriched with a large-scale analysis of liver tissues using shotgun proteomics to characterize protein networks from tumor microenvironment whose remodeling is responsible for extracellular activation of TGF-β. The trajectories and upstream regulators of the final model will be analyzed with symbolic model checking techniques and abstract interpretation combined with causality analysis. Candidates will be classified with semantic-based approaches and symbolic bi-clustering technics. All efforts must ultimately converge to experimental validations of hypotheses and we will use our hepatic cellular models (HCC cell lines and hepatic stellate cells) to screen inhibitors on the behaviors of TGF-β signal.

The expected results are the first model of extracellular and intracellular TGF-β system that might permit to analyze the behaviors of TGF-β activity during the course of liver tumor progression and to identify new biomarkers and potential therapeutic targets.

7.1.5. VeriAMOS

Title: Verification of Abstract Machines for Operating Systems

Type: ANR générique 2018

Defi: Société de l'information et de la communication

Instrument: ANR grant

Duration: January 2019 - December 2022

Coordinator: Inria Paris (France)

Others partners: LIP6 (France), IRISA (France), UGA (France)

Inria contact: Xavier Rival

Abstract: Operating System (OS) programming is notoriously difficult and error prone. Moreover, OS bugs can have a serious impact on the functioning of computer systems. Yet, the verification of OSes is still mostly an open problem, and has only been done using user-assisted approaches that require a huge amount of human intervention. The VeriAMOS proposal relies on a novel approach to automatically and fully verifying OS services, that combines Domain Specific Languages (DSLs) and automatic static analysis. In this approach, DSLs provide language abstraction and let users express complex policies in high-level simple code. This code is later compiled into low level C code, to be executed on an abstract machine. Last, the automatic static analysis verifies structural and robustness properties on the abstract machine and generated code. We will apply this approach to the automatic, full verification of input/output schedulers for modern supports like SSDs.

7.2. European Initiatives

7.2.1. FP7 & H2020 Projects

Type: IDEAS

Defi:

Instrument: ERC Proof of Concept Grant 2018

Objectif: Static Analysis for the VErification of Spreadsheets

Duration: January 2019 - June 2020

Coordinator: Inria (France)

Partner: None

Inria contact: Xavier Rival

Abstract: Spreadsheet applications (such as Microsoft Excel + VBA) are heavily used in a wide range of application domains including engineering, finance, management, statistics and health. However, they do not ensure robustness properties, thus spreadsheet errors are common and potentially costly. According to estimates, the annual cost of spreadsheet errors is around 7 billion dollars. For instance, in 2013, a series of spreadsheet errors at JPMorgan incurred 6 billion dollars trading losses. Yet, expert reports estimate about 90 % of the spreadsheets contain errors. The MemCAD ERC StG project opened the way to novel formal analysis techniques for spreadsheet applications. We propose to leverage these results into a toolbox able to safely *verify*, *optimize* and *maintain* spreadsheets, so as to reduce the likelihood of spreadsheet disasters. This toolbox will be commercialized by the startup MATRIXLEAD.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

7.3.1.1. Internships

Jérôme Feret has supervised the M1 Internship of Aurélie Faure de Pebeyre (AIV Master) and the M2 Internship of Albin Salazar (AIV Master).

Xavier Rival is supervising M1 Internships of Guillaume Reboullet and of Luc Chabassier (M1 at DIENS).

Vincent Danos supervised interns Raja Ben Ali and Jaime Aujaud (L2 Epitech).

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

- Jérôme Feret is a guest member of the Steering Committee of the Conference on Computational Methods in Systems Biology (CMSB).
- Jérôme Feret is a member of the Steering Committee of the Workshop on Static Analysis and Systems Biology (SASB).
- Xavier Rival organized the 60th meeting of the IFIP Working Group 2.4 held in Dijon, in July 2018.
- Xavier Rival is a member of the Steering Committee of the Static Analysis Symposium (SAS).
- Xavier Rival is a member of the Steering Committee of the Workshop on Tools for Automatic Program Analysis (TAPAS).

8.1.2. Scientific Events Selection

8.1.2.1. Chair of Conference Program Committees

- Xavier Rival served as Chair of the Artifact Evaluation Committee of SAS 2018 (Static Analysis Symposium).

8.1.2.2. Member of the Conference Program Committees

- Jérôme Feret served as a Member of the Program Committee of LICS 2018 (Logic in Computer Science).
- Jérôme Feret served as a Member of the Program Committee of VEMDP 2018 (Verification of Engineered Molecular Devices and Programs).
- Jérôme Feret served as a Member of the Program Committee of SASB 2018 (Static Analysis and Systems Biology Workshop).
- Jérôme Feret served as a Member of the Program Committee of SAS 2018 (Static Analysis Symposium).
- Jérôme Feret served as a Member of the Program Committee of CMSB 2018 (Computational Methods in Systems Biology).
- Jérôme Feret served as a Member of the Program Committee of VMCAI 2019 (Verification, Model Checking, and Abstract Interpretation)
- Jérôme Feret is serving as a Member of the Program Committee of CIBCB 2019 (Computational Intelligence in Bioinformatics and Computational Biology).
- érôme Feret is serving as a Member of the Program Committee of HSB 2019 (Hybrid Systems and Biology).
- Jérôme Feret is serving as a Member of the Program Committee of CMSB 2019 (Computational Methods in Systems Biology).
- Xavier Rival served as a Member of the Program Committee of SAS 2018 (Static Analysis Symposium).
- Xavier Rival served as a Member of the Program Committee of Web Design, Analysis, Programming and Implementation of the WWW'18 Conference.
- Xavier Rival served as a Member of the Extended Review Committee of PLDI 2018 (Programming Languages Design and Implementation).
- Xavier Rival served as a Member of the Program Committee of APLAS 2018 (Asian Programming Languages And Systems Symposium).
- Xavier Rival is serving as a Member of the Committee of POPL 2020 (Principles of Programming Languages).
- Cezara Drăgoi served as a member of the Program Committee of Computer Aided Verification CAV'18.
- Cezara Drăgoi served as a member of the Program Committee of VMCAI 2019 (Verification, Model Checking, and Abstract Interpretation).
- Cezara Drăgoi served as a member of the Program Committee of NETYS 2018.

8.1.2.3. Reviewer

- Jérôme Feret served as Reviewer for ARSBM 2018 (Automated Reasoning for Systems Biology and Medicine).

8.1.3. Journal

8.1.3.1. Member of the Editorial Boards

- Jérôme Feret serves as a Member of the Editorial Board of the Frontiers in Genetics journal and the Open Journal of Modeling and Simulation.

- Jérôme Feret serves as co-Editor of an Issue of the Theoretical Computer Science journal, that is composed of papers from SASB 2016, and is expected to appear in 2019.
- Jérôme Feret serves as co-Editor of an Issue of the IEEE/ACM Transactions on Computational Biology and Bioinformatics, that is composed of papers from CMSB 2017, and is expected to appear in 2019.
- Xavier Rival serves as Editor of an Issue of the Formal Methods in System Design Journal, that is composed of a selection of papers from SAS 2016, and appeared in 2018.

8.1.3.2. Reviewer - Reviewing Activities

- Jérôme Feret served as a Reviewer for NACO (Natural Computing).
- Jérôme Feret served as a Reviewer for ACS Synthetic Biology.
- Jérôme Feret served as a Reviewer for TCS (Theoretical Computer Sciences).
- Jérôme Feret served as a Reviewer for TCBB (IEEE/ACM Transactions on Computational Biology and Bioinformatics).
- Jérôme Feret served as a Reviewer for IEEE Transactions on Reliability.
- Xavier Rival served as a Reviewer for ACM TOPLAS (Transactions On Programming Languages and Systems).
- Xavier Rival served as a Reviewer for ACM TOPS (Transactions On Privacy and Security).

8.1.4. Invited Talks

- Cezara Drăgoi was invited to give a talk at the workshop on Verification of Distributed Systems, Essaouira, Morocco, 2018.
- Cezara Drăgoi was invited to give a talk at the Dagstuhl workshop no 18211 on Formal Methods and Fault-Tolerant Distributed Computing: Forging an Alliance.

8.1.5. Leadership within the Scientific Community

Xavier Rival is a member of the IFIP Working Group 2.4 on Software implementation technology.

8.1.6. Scientific Expertise

- Cezara Drăgoi has participated to the recruitment committee for an assistant professor in the Department of Computer Science of École normale supérieure 2018.
- Jérôme Feret served as a external Reviewer for research program PRIM 2017 (funded by MIUR, the Italian Ministry for Education, University and Research).
- Jérôme Feret has participated to the recruitment committee for an assistant professor in Paris-Diderot University 2018.
- Xavier Rival chaired the Hiring Committee for an Assistant Professor position (Tenure track position, Gaspard Monge Chair) at École Polytechnique in 2018.

8.1.7. Research Administration

- Jérôme Feret and Xavier Rival are members of the Laboratory Council of DIENS.
- Jérôme Feret is member of PhD Review Committee (CSD) of Inria Paris.
- Jérôme Feret is deputy head of study of the Department of Computer Science of École normale supérieure.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence:

- Marc Chevalier, Mathematics, 40h, L1, FDV Bachelor program (Frontiers in Life Sciences (FdV)), Université Paris-Descartes, France.
- Jérôme Feret and Xavier Rival (lectures), and Marc Chevalier (tutorials), “Semantics and Application to Verification”, 36h, L3, at École Normale Supérieure, France.
- Xavier Rival, “Introduction to Static Analysis”, 8h, L3, at École des Mines de Paris, France.
- Xavier Rival, “Programmation Avancée”, 18h, L3, at École Polytechnique, France.

Master:

- Xavier Rival, “Introduction to Static Analysis”, 24h, Internet of Things Master (retraining curriculum, EXED), France.
- Xavier Rival, “Protocol Safety and Verification”, 12h, M2, Advanced Communication Networks Master, France.
- Cezara Drăgoi, Jérôme Feret, Antoine Miné, and Xavier Rival, “Abstract Interpretation: application to verification and static analysis”, 72h, M2. Parisian Master of Research in Computer Science (MPRI), France.
- Vincent Danos and Jérôme Feret (with Jean Krivine), Computational Biology, 28h, M1. Interdisciplinary Approaches to Life Science (AIV), Master Program, Université Paris-Descartes, France.

Doctorat:

- Jérôme Feret, “Intrinsic Coarse-Graining of Models of Signalling pathways", 1h30, aDVANCES IN SYSTEMS AND SYNTHETIC BIOLOGY Modelling Complex Biological Systems in the Context of Genomics Thematic Research School, March 2018, Évry, France.

8.2.2. Supervision

- PhD in progress: Marc Chevalier, Static analysis of Security Properties in Critical Embedded Software. started in 2017 and supervised by Jérôme Feret
- PhD in progress: Hugo Illous, Relational Shape Abstraction Based on Separation Logic, started in 2015 and supervised by Xavier Rival and Matthieu Lemerre (CEA)
- PhD in progress: Olivier Nicole, Verification of micro-kernels, started in 2018 and supervised by Xavier Rival and Matthieu Lemerre (CEA)
- PhD defended: Huisong Li, Disjunctive Shape Abstraction for Shared Data-Structures, started in 2014 and supervised by Xavier Rival
- PhD defended: Jiangchao Liu, Static Analysis for Numeric and Structural Properties of Array Contents, started in 2014 and supervised by Xavier Rival

8.2.3. Juries

- Xavier Rival served as a Reviewer in the Jury of the PhD of Ahmad Salim Al-Sibahi (Defense planned for the 11th of January, 2018).

8.3. Popularization

8.3.1. Internal or external Inria responsibilities

- Xavier Rival is member of the “Bureau du comité des projets” since October 2018.
- Xavier Rival is Chair of the Hiring Committee for Inria researchers at the center of Paris (CRCN) for 2019.

9. 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] A. BOUAJJANI, C. DRAGOI, C. ENEA, M. SIGHIREANU. *On inter-procedural analysis of programs with lists and data*, in "Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2011, San Jose, CA, USA, June 4-8, 2011", 2011, pp. 578–589 [*DOI :* 10.1145/1993498.1993566], https://dl.acm.org/citation.cfm?id=1993498.1993566

[4] 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

[5] 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

[6] 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

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

[8] 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

Doctoral Dissertations and Habilitation Theses

[9] H. LI. *Shape Abstractions with Support for Sharing and Disjunctions*, ENS Paris - Ecole Normale Supérieure de Paris ; PSL University, January 2018, https://hal.archives-ouvertes.fr/tel-01963082

[10] J. LIU. *Static Analysis on Numeric and Structural Properties of Array Contents*, ENS Paris ; PSL University, February 2018, https://hal.archives-ouvertes.fr/tel-01963108

Articles in International Peer-Reviewed Journals

[11] P. BOUTILLIER, M. MAASHA, X. LI, H. F. MEDINA-ABARCA, J. KRIVINE, J. FERET, I. CRISTESCU, A. G. FORBES, W. FONTANA. *The Kappa platform for rule-based modeling*, in "Bioinformatics", July 2018, vol. 34, n° 13, pp. i583-i592 [*DOI :* 10.1093/BIOINFORMATICS/BTY272], https://hal.inria.fr/hal-01962663

[12] F. DAHLQVIST, A. SILVA, V. DANOS, I. GARNIER. *Borel Kernels and their Approximation, Categorically*, in "Electronic Notes in Theoretical Computer Science", December 2018, vol. 341, pp. 91-119, https://hal.archives-ouvertes.fr/hal-01976416

[13] J. FERET, K. Q. LÝ. *Local Traces: An Over-Approximation of the Behavior of the Proteins in Rule-Based Models*, in "IEEE/ACM Transactions on Computational Biology and Bioinformatics", July 2018, vol. 15, n° 4, pp. 1124-1137, https://hal.inria.fr/hal-01967635

[14] S.-W. KIM, X. RIVAL, S. RYU. *A Theoretical Foundation of Sensitivity in an Abstract Interpretation Framework*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", August 2018, vol. 40, n° 3, pp. 1-44, https://hal.archives-ouvertes.fr/hal-01963069

[15] J. LIU, L. CHEN, X. RIVAL. *Automatic Verification of Embedded System Code Manipulating Dynamic Structures Stored in Contiguous Regions*, in "IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems", November 2018, vol. 37, n° 11, pp. 2311-2322, https://hal.archives-ouvertes.fr/hal-01963049

[16] G. TERRADOT, A. BEICA, A. Y. WEISSE, V. DANOS. *Survival of the Fattest: Evolutionary Trade-offs in Cellular Resource Storage*, in "Electronic Notes in Theoretical Computer Science", April 2018, vol. 335, pp. 91-112 [*DOI :* 10.1016/J.ENTCS.2018.03.010], https://hal.archives-ouvertes.fr/hal-01976385

[17] P. THOMAS, G. TERRADOT, V. DANOS, A. Y. WEISSE. *Sources, propagation and consequences of stochasticity in cellular growth*, in "Nature Communications", December 2018, vol. 9, n° 1 [*DOI :* 10.1038/S41467-018-06912-9], https://hal.archives-ouvertes.fr/hal-01976406

[18] W. WAITES, M. CAVALIERE, E. CACHAT, V. DANOS, J. A. DAVIES. *An Information-Theoretic Measure for Patterning in Epithelial Tissues*, in "IEEE Access", 2018, vol. 6, pp. 40302-40312 [*DOI :* 10.1109/ACCESS.2018.2853624], https://hal.archives-ouvertes.fr/hal-01976389

[19] W. WAITES, G. MISIRLI, M. CAVALIERE, V. DANOS, A. WIPAT. *A Genetic Circuit Compiler: Generating Combinatorial Genetic Circuits with Web Semantics and Inference*, in "ACS Synthetic Biology", November 2018, vol. 7, n° 12, pp. 2812-2823, https://hal.archives-ouvertes.fr/hal-01985802

International Conferences with Proceedings

[20] A. BEICA, J. FERET, T. PETROV. *Tropical Abstraction of Biochemical Reaction Networks with Guarantees*, in "SASB'18 - Static Analysis in Systems Biology, affiliated with Static Analysis Symposium", Freiburg, Germany, August 2018, https://hal.inria.fr/hal-01962674

[21] P. BOUTILLIER, F. CAMPORESI, J. COQUET, J. FERET, K. Q. LÝ, N. THÉRET, P. VIGNET. *KaSa: A Static Analyzer for Kappa*, in "CMSB 2018 - 16th International Conference on Computational Methods in Systems Biology", Brno, Czech Republic, M. ČEŠKA, D. ŠAFRÁNEK (editors), LNCS, Springer Verlag, September 2018, vol. 11095, pp. 285-291 [*DOI :* 10.1007/978-3-319-99429-1_17], https://hal-univ-rennes1.archives-ouvertes.fr/hal-01888951

[22] P. BOUTILLIER, J. FERET, A. FAURE DE PEBEYRE. *Proving the absence of unbounded polymers in rule-based models*, in "Static Analysis and Systems Biology 2018", Freiburg im Breisgau, Germany, August 2018, https://hal.inria.fr/hal-01967632

[23] T. SUZANNE, A. MINÉ. *Relational Thread-Modular Abstract Interpretation Under Relaxed Memory Models*, in "APLAS 2018 - 16th Asian Symposium on Programming Languages and Systems", Wellington, New Zealand, Lecture Notes in Computer Science, December 2018, vol. 11275, pp. 109-128 [*DOI :* 10.1007/978-3-030-02768-1_6], https://hal.inria.fr/hal-01953358

Other Publications

[24] A. DAMIAN, C. DRAGOI, A. MILITARU, J. WIDDER. *Communication-closed asynchronous protocols*, January 2019, working paper or preprint, https://hal.inria.fr/hal-01991415

References in notes

[25] A. BEICA, V. DANOS. *Synchronous Balanced Analysis*, in "Proceedings of the International Workshop on Hybrid Systems Biology", Springer-Verlag, Berlin, Germany, September 2016, pp. 85-94, https://hal.archives-ouvertes.fr/hal-01974696

[26] 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

[27] 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

[28] Y. KO, X. RIVAL, S. RYU. *Weakly Sensitive Analysis for Unbounded Iteration over JavaScript Objects*, in "APLAS 2017 - 15th Asian Symposium on Programming Languages and Systems", Suzhou, China, LNCS, Springer, November 2017, vol. 10695, pp. 148-168 [*DOI :* 10.1007/978-3-319-71237-6_8], https://hal.inria.fr/hal-01648680

[29] H. LI, F. BÉRENGER, B.-Y. E. CHANG, X. RIVAL. *Semantic-Directed Clumping of Disjunctive Abstract States* *, in "POPL 2017 - 44th ACM SIGPLAN Symposium on Principles of Programming Languages", Paris, France, ACM, January 2017, vol. 52, n^{o} 1, pp. 32-45 [*DOI :* 10.1145/3009837.3009881], https://hal.inria.fr/hal-01648679

[30] A. Y. WEISSE, D. A. OYARZUN, V. DANOS, P. S. SWAIN. *Mechanistic links between cellular trade-offs, gene expression, and growth*, in "Proceedings of the National Academy of Sciences of the United States of America ", March 2015, vol. 112, n^{o} 9, pp. E1038-E1047 [*DOI :* 10.1073/PNAS.1416533112], https://hal.archives-ouvertes.fr/hal-01976394