IN PARTNERSHIP WITH:
Université Rennes 1

École normale supérieure de Rennes

Activity Report 2016

Project-Team CELTIQUE

Software certification with semantic analysis

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Proofs and Verification

Table of contents

Project-Team CELTIQUE

Creation of the Project-Team: 2009 July 01

Keywords:

Computer Science and Digital Science:

2.1. - Programming Languages
2.1.1. - Semantics of programming languages
2.1.2. - Object-oriented programming
2.1.3. - Functional programming
2.1.9. - Dynamic languages
2.2. - Compilation
2.2.1. - Static analysis
2.2.2. - Memory models
2.4. - Verification, reliability, certification
2.4.1. - Analysis
2.4.2. - Model-checking
2.4.3. - Proofs
4. - Security and privacy
4.5. - Formal methods for security

Other Research Topics and Application Domains:

6.1. - Software industry
6.1.1. - Software engineering
6.6. - Embedded systems

1. Members

Research Scientists

Thomas Jensen [Team leader, Inria, Senior Researcher, HDR]
Frédéric Besson [Inria, Researcher]
Alan Schmitt [Inria, Senior Researcher, HDR]

Faculty Members

Sandrine Blazy [Univ. Rennes I, Professor, HDR]
David Cachera [ENS Rennes, Associate Professor, HDR]
Delphine Demange [Univ. Rennes I, Associate Professor]
Thomas Genet [Univ. Rennes I, Associate Professor, HDR]
Serguei Lenglet [Univ. Lorraine, Associate Professor]
David Pichardie [ENS Rennes, Professor, HDR]
Charlotte Truchet [Univ. Nantes, Associate Professor]

Engineers

Laurent Guillo [CNRS, Senior engineer]
Pierre Wilke [Inria, until July 2016]

PhD Students

Oana Andreescu [Prove & Run, until Sep 2016, granted by CIFRE]
Martin Bodin [Inria, until Nov 2016]

Pauline Bolignano [Prove & Run, granted by CIFRE]
David Buhler [CEA]
Gurvan Cabon [Inria]
Alexandre Dang [Inria, from Oct 2016]
Yon Fernandez de Retana [Univ. Rennes I]
Julien Lepiller [Inria, from Oct 2016]
Florent Saudel [Amossys, granted by CIFRE]
Alix Trieu [Univ. Rennes I]
Yannick Zakowski [ENS Rennes]

Post-Doctoral Fellow
Cyrille Wiedling [Inria]

Visiting Scientist
Ahmad Salim Al-Sibahi [IT University of Copenhagen, from Sep 2016]

Administrative Assistant
Lydie Mabil [Inria]

Others
Aurele Barriere [Normale Sup Rennes, Internship, from May 2016 until Jul 2016]
Thomas Wood [Inria, PhD Student, from Oct 2016]

2. Overall Objectives

2.1. Project overview

The overall goal of the CELTIQUE project is to improve the security and reliability of software with semantics-based modeling, analysis and certification techniques. To achieve this goal, the project conducts work on improving semantic description and analysis techniques, as well as work on using proof assistants (most notably Coq) to develop and prove properties of these techniques. We are applying such techniques to a variety of source languages, including Java, C, and JavaScript. We also study how these techniques apply to low-level languages, and how they can be combined with certified compilation. The CompCert certified compiler and its intermediate representations are used for much of our work on semantic modeling and analysis of C and lower-level representations.

The semantic analyses extract approximate but sound descriptions of software behaviour from which a proof of safety or security can be constructed. The analyses of interest include numerical data flow analysis, control flow analysis for higher-order languages, alias and points-to analysis for heap structure manipulation. In particular, we have designed several analyses for information flow control, aimed at computing attacker knowledge and detecting side channels.

We work with three application domains: Java software for small devices (in particular smart cards and mobile telephones), embedded C programs, and web applications.

CELTIQUE is a joint project with the CNRS, the University of Rennes 1 and ENS Rennes.

3. New Software and Platforms

3.1. JSCert

Certified JavaScript
FUNCTIONAL DESCRIPTION

The JSCert project aims to develop a formal understanding of the JavaScript programming language. JSCert itself is a mechanised specification of JavaScript, written in the Coq proof assistant, which closely follows the ECMAScript 5 English standard. JSRef is a reference interpreter for JavaScript in OCaml , which has been proved correct with respect to JSCert and tested with the Test 262 test suite.

- Participants: Martin Bodin and Alan Schmitt
- Partner: Imperial College London
- Contact: Alan Schmitt
- URL: http://jscert.org/

3.2. Javalib

FUNCTIONAL DESCRIPTION

Javalib is an efficient library to parse Java .class files into OCaml data structures, thus enabling the OCaml programmer to extract information from class files, to manipulate and to generate valid .class files.

- Participants: Frederic Besson, David Pichardie, Pierre Vittet, Laurent Guillo, Laurent Hubert, Tiphaine Turpin and Nicolas Barre
- Contact: Frederic Besson
- URL: http://sawja.inria.fr/

3.3. SAWJA

Static Analysis Workshop for Java
KEYWORDS: Security - Software - Code review - Smart card
SCIENTIFIC DESCRIPTION

Sawja is a library written in OCaml, relying on Javalib to provide a high level representation of Java bytecode programs. It name comes from Static Analysis Workshop for JAva. Whereas Javalib is dedicated to isolated classes, Sawja handles bytecode programs with their class hierarchy and with control flow algorithms.

Moreover, Sawja provides some stackless intermediate representations of code, called JBir and A3Bir. The transformation algorithm, common to these representations, has been formalized and proved to be semantics-preserving.

See also the web page http://sawja.inria.fr/.

Version: 1.5

Programming language: Ocaml
FUNCTIONAL DESCRIPTION

Sawja is a toolbox for developing static analysis of Java code in bytecode format. Sawja provides advanced algorithms for reconstructing high-level programme representations. The SawjaCard tool dedicated to JavaCard is based on the Sawja infrastructure and automatically validates the security guidelines issued by AFSCM (http://www.afscm.org/). SawjaCard can automate the code audit process and automatic verification of functional properties.

- Participants: Frederic Besson, David Pichardie and Laurent Guillo
- Partners: CNRS - ENS Rennes
- Contact: Frederic Besson
- URL: http://sawja.inria.fr/

3.4. Timbuk

KEYWORDS: Proof - Ocaml - Program verification - Tree Automata
FUNCTIONAL DESCRIPTION

Timbuk is a collection of tools for achieving proofs of reachability over Term Rewriting Systems and for manipulating Tree Automata (bottom-up non-deterministic finite tree automata)

- Participant: Thomas Genet
- Contact: Thomas Genet
- URL: http://www.irisa.fr/celtique/genet/timbuk/

3.5. CompCertSSA

KEYWORDS: Verified compilation - Single Static Assignment form - Optimization - Coq - OCaml

FUNCTIONAL DESCRIPTION

CompCertSSA is built on top of the C CompCert verified compiler, by adding a SSA-based middle-end (conversion to SSA, SSA-based optimizations, destruction of SSA).

Notably, the middle-end features:

- new important optimizations (Sparse Conditional Constant Propagation, and a coalescing phase on Conventional SSA)
- a generic dominance-based proof framework that rationalizes the proof process
- improved performance regarding compilation time

It is verified in the Coq proof assistant.

- Participant: Delphine Demange, David Pichardie, Yon Fernandez de Retana, Leo Stefanesco
- Contact: Delphine Demange
- URL: http://compcertssa.gforge.inria.fr/

4. New Results

4.1. Monitoring attacker knowledge with information flow analysis

Participants: Thomas Jensen, Frédéric Besson.

Motivated by the problem of stateless web tracking (fingerprinting) we have investigated a novel approach to hybrid information flow monitoring by tracking the knowledge that an attacker can learn about secrets during a program execution. We have proposed a general framework for combining static and dynamic information flow analysis, based on a precise representation of attacker knowledge. This hybrid analysis computes a precise description of what an attacker learns about the initial configuration (and in particular the secret part of it) by observing a specific output. An interesting feature of this knowledge-based information flow analysis is that it can be used to improve other information flow control mechanisms, such as no-sensitive upgrade. The whole framework is accompanied by a formalisation of the theory in the Coq proof assistant [18].

4.2. Semantic analysis of functional specifications of system software

Participants: Thomas Jensen, Oana Andreescu, Pauline Bolignano.

We have developed a static analysis for correlating input and output values in functional specifications, written in a functional, strongly typed, high-level specification formalism developed by the SME Prove & Run. In the context of interactive formal verification of complex systems, much effort is spent on proving the preservation of the system invariants. However, most operations have a localized effect on the system. Identifying correlations (in particular equalities) between input and output can substantially ease the proof burden for the programmer. Our correlation analysis is a flow-sensitive interprocedural analysis that handles arrays, structures and variant data types, and which computes a conservative approximation of the equality between sub-structures of input and of output fragments [27]. In a separate strand of work, we have used abstraction-based techniques for structuring and simplifying the proof of simulation between a high-level and a low-level specification of memory management algorithms in a hypervisor [22]. Both strands of work was carried out and validated on system software (a micro-kernel and a hypervisor) developed using the formal approach defined by Prove & Run.

4.3. Certified Static Analyses

4.3.1. Certified Semantics and Analyses for JavaScript

Participants: Martin Bodin, Gurvan Cabon, Thomas Jensen, Alan Schmitt.

We have continued our work on the certification of the semantics of JavaScript and of analyses for JavaScript on three different fronts.

First, on the language side, we have developed at tool in collaboration with Arthur Charguéraud (Inria Saclay) and Thomas Wood (Imperial College) to interactively explore the specification of JavaScript. More precisely, we have written a compiler for a subset of OCaml to a subset of JavaScript that generates an interpreter that can be executed step by step, inspecting both the state of the interpreted program but also the state of the interpreter. We have used this compiler on the JavaScript interpreter extracted from our Coq semantics of JavaScript. The resulting tool is available here and a demo can be run here. The tool has been presented to the Ecma TC39 committee in charge of standardizing JavaScript. We are currently identifying the improvements required to make it useful for the standardization process.

Second, Bodin, Schmitt, and Jensen have designed an abstract domain based on separation logic to faithfully abstract JavaScript heaps. This domain is able to capture interlinked dynamic and extensible objects, a central feature of the JavaScript memory model. In addition, we have introduced the notion of *membranes* that let us correctly define abstractions in a way that is compatible both with separation logic and abstract interpretation. As an extension of last year's work [32], this approach is globally correct as soon as each rule is independently proven correct. This result illustrates the robustness of our approach to define certified abstract semantics based on pretty-big-step semantics. This work has not yet been published.

Third, Cabon and Schmitt are developing a framework to automatically derive an information-flow tracking semantics from a pretty-big-step semantics. We have manually shown the approach works for complex examples, and are currently proving it in Coq. This work is submitted for publication.

4.3.2. Certified Analyses for C and lower-level programs

Participants: Sandrine Blazy, David Pichardie, Alix Trieu.

We have continued our work on the static analyzer Verasco [37], based on abstract interpretation and operating over most of the ISO C 1999 language (excluding recursion and dynamic allocation). Verasco establishes the absence of run-time errors in the analyzed programs. It enjoys a modular architecture that supports the extensible combination of multiple abstract domains. We have extended the memory abstract domain (that takes as argument any standard numerical abstract domain), so that it finely tracks properties about memory contents, taking into account union types, pointer arithmetic and type casts [19]. This memory domain is implemented and verified inside the Coq proof assistant with respect to the CompCert compiler memory model.

Motivated by applications to security and high efficiency, we are reusing the Verasco static analyzer and the CompCert compiler in order to design a lightweight and automated methodology for validating on low-level intermediate representations the results of a source-level static analysis. Our methodology relies on two main ingredients: a relative-safety checker, an instance of a relational verifier which proves that a program is safer than another, and a transformation of programs into defensive form which verifies the analysis results at runtime.

4.4. Certified Compilation

Participants: Sandrine Blazy, Frédéric Besson, Pierre Wilke, Alexandre Dang.

The COMPCERT C compiler provides the formal guarantee that the observable behaviour of the compiled code improves on the observable behaviour of the source code. A first limitation of this guarantee is that if the source code goes wrong, i.e. does not have a well-defined behaviour, any compiled code is compliant. Another limitation is that COMPCERT 's notion of observable behaviour is restricted to IO events.

Over the past years, we have developed the semantics theory so that unlike CompCert but like Gcc, the binary representation of pointers can be manipulated much like integers and where memory is a finite resource. We have now a formally verified C compiler, CompCertS, which is essentially the CompCert compiler, albeit with a stronger formal guarantee. The semantics preservation theorem applies to a wider class of existing C programs and, therefore, their compiled version benefits from the formal guarantee of CompCertS. CompCertS preserves not only the observable behaviour of programs but also ensures that the memory consumption is preserved by the compiler. As a result, we have the formal guarantee that the compiled code requires no more memory than the source code. This ensures that the absence of stack-overflows is preserved by compilation.

The whole proof of CompCertS represents a significant proof-effort and the details can be found in Pierre Wilke's PhD thesis [39].

CompCertS also implements the Portable Software Fault Isolation approach pioneered by Kroll *et al.* [38]. The advantage of CompCertS is that the masking operation of pointers has a defined semantics and can therefore be directly reasoned about.

4.5. Mechanical Verification of SSA-based Compilation Techniques

Participants: Delphine Demange, Yon Fernandez de Retana, David Pichardie.

We have continued our work on the mechanical verification of SSA-based compilation techniques [30], [31], [36].

A crucial phase for efficient machine code generation is the destruction of a middle-end SSA-like IR. To this end, we have studied a variant of SSA, namely the Conventional SSA form, which simplifies the destruction back to non-SSA code (i.e. at the exit point of the middle-end). This had long remained a difficult problem, even in a non-verified environment. We formally defined and proved the properties of the generation of Conventional SSA. Finally, we implemented and proved correct a coalescing destruction of the Conventional SSA form, à la Boissinot et al. [33], where variables can be coalesced according to a refined notion of interference. Our CSSA-based, coalescing destruction allows us to coalesce more than 99% of introduced copies, on average, and leads to encouraging results concerning spilling and reloading during post-SSA allocation. This work has been published in [24].

4.6. Semantics for shared-memory concurrency

Participants: Gurvan Cabon, David Cachera, David Pichardie.

Modern multicore processor architectures and compilers of shared-memory concurrent programming languages provide only weak memory consistency guarantees. A *memory model* specifies which write action can be seen by a read action between concurrent threads.

In a previous work on the Java memory model [35], we defined in an axiomatic style, a memory model where we embed the reorderings of memory accesses directly in the semantics, so that formalizing optimizations and their correctness proof is easier.

This year, following a similar approach, we have studied the RMO (Relaxed- Memory Order) model. More precisely, we defined a new multibuffer operational semantics with write and read buffers. We also introduced an intermediate semantics inspired from Boudol et al. [34], where actions are reordered within a single pipeline. Finally, another model formalizes the reordering semantics in an axiomatic way. We fully proved the equivalence between the first two models and present a methodology for the remaining part. This work has been published in an international workshop [23].

4.7. Static analysis of functional programs using tree automata and term rewriting

Participant: Thomas Genet.

We develop a specific theory and the related tools for analyzing programs whose semantics is defined using term rewriting systems. The analysis principle is based on regular approximations of infinite sets of terms reachable by rewriting. Regular tree languages are (possibly) infinite languages which can be finitely represented using tree automata. To over-approximate sets of reachable terms, the tools we develop use the Tree Automata Completion (TAC) algorithm to compute a tree automaton recognizing a superset of all reachable terms. This over-approximation is then used to prove properties on the program by showing that some "bad" terms, encoding dangerous or problematic configurations, are not in the superset and thus not reachable. This is a specific form of, so-called, Regular Tree Model Checking. In [16], we have shown two results. The first result is a precision result guaranteeing that, for most of term rewriting systems known to have a regular set of reachable terms, TAC always compute it in an exact way. The second result shows that tree automata completion can be applied to functional programs to over-approximate their image. In particular, we have shown that tree automata completion computes a safe over-approximation of the image of any first-order, purely functional, complete and terminating program. Now, our first next objective is to demonstrate the accuracy of those regular approximations to perform lightweight formal verification of functional programs. The second objective is to lift those results to higher-order purely functional programs.

5. Partnerships and Cooperations

5.1. Regional Initiatives

5.1.1. Labex COMIN Labs Seccloud project

Participants: Frédéric Besson, Thomas Jensen, Alan Schmitt, Thomas Genet, Martin Bodin, Gurvan Cabon.

The SecCloud project, started in 2012, will provide a comprehensive language-based approach to the definition, analysis and implementation of secure applications developed using Javascript and similar languages. Our high level objectives is to enhance the security of devices (PCs, smartphones, ect.) on which Javascript applications can be downloaded, hence on client-side security in the context of the Cloud. We will achieve this by focusing on three related issues: declarative security properties and policies for client-side applications, static and dynamic analysis of web scripting programming languages, and multi-level information flow monitoring.

This is a joint project with Supelec Rennes and Ecole des Mines de Nantes.

5.2. National Initiatives

5.2.1. The ANR VERASCO project

Participants: Sandrine Blazy, Delphine Demange, David Pichardie.

Static program analysis, Certified static analysis

The VERASCO project (2012–06/2016) is funded by the call ISN 2011, a program of the Agence Nationale de la Recherche. It investigates the formal verification of static analyzers and of compilers, two families of tools that play a crucial role in the development and validation of critical embedded software. It is a joint project with the Inria teams ABSTRACTION, GALLIUM, The VERIMAG laboratory and the Airbus company.

5.2.2. The ANR AnaStaSec project

Participants: Frédéric Besson, Sandrine Blazy, Thomas Jensen, Alexandre Dang, Julien Lepiller.

Static program analysis, Security, Secure compilation

The AnaStaSec project (2015–2018) aims at ensuring security properties of embedded critical systems using static analysis and security enhancing compiler techniques. The case studies are airborne embedded software with ground communication capabilities. The Celtique project focuses on software fault isolation which is a compiler technology to ensure by construction a strong segregation of tasks.

This is a joint project with the Inria teams ANTIQUE and PROSECCO, CEA-LIST, TrustInSoft, AMOSSYS and Airbus Group.

5.2.3. *The ANR Binsec project*

Participants: Frédéric Besson, Sandrine Blazy, Pierre Wilke, Julien Lepiller.

Binary code, Static program analysis

The Binsec project (2013–2017) is funded by the call ISN 2012, a program of the Agence Nationale de la Recherche. The goal of the BINSEC project is to develop static analysis techniques and tools for performing automatic security analyses of binary code. We target two main applicative domains: vulnerability analysis and virus detection.

Binsec is a joint project with the Inria CARTE team, CEA LIS, VERIMAG and EADS IW.

5.2.4. *The ANR MALTHY project*

Participant: David Cachera.

The MALTHY project, funded by ANR in the program INS 2013, aims at advancing the state-of-the-art in real-time and hybrid model checking by applying advanced methods and tools from linear algebra and algebraic geometry. MALTHY is coordinated by VERIMAG, involving CEA-LIST, Inria Rennes (Tamis and Celtique), Inria Saclay (MAXPLUS) and VISEO/Object Direct.

5.2.5. *The ANR AJACS project*

Participants: Martin Bodin, Gurvan Cabon, Thomas Jensen, Alan Schmitt.

The goal of the AJACS project is to provide strong security and privacy guarantees on the client side for web application scripts. To this end, we propose to define a mechanized semantics of the full JavaScript language, the most widely used language for the Web. We then propose to develop and prove correct analyses for JavaScript programs, in particular information flow analyses that guarantee no secret information is leaked to malicious parties. The definition of sub-languages of JavaScript, with certified compilation techniques targeting them, will allow us to derive more precise analyses. Finally, we propose to design and certify security and privacy enforcement mechanisms for web applications, including the APIs used to program real-world applications.

The project partners include the following Inria teams: Celtique, Indes, Prosecco, and Toccata; it also involves researchers from Imperial College as external collaborators. The project runs from December 2014 to June 2018.

5.2.6. *The ANR DISCOVER project*

Participants: Sandrine Blazy, Delphine Demange, Thomas Jensen, David Pichardie, Yon Fernandez de Retana.

The DISCOVER project project aims at leveraging recent foundational work on formal verification and proof assistants to design, implement and verify compilation techniques used for high-level concurrent and managed programming languages. The ultimate goal of DISCOVER is to devise new formalisms and proof techniques able to scale to the mechanized correctness proof of a compiler involving a rich class of optimizations, leading to efficient and scalable applications, written in higher-level languages than those currently handled by cutting-edge verified compilers.

In the light of recent work in optimizations techniques used in production compilers of high-level languages, control-flow-graph based intermediate representations seems too rigid. Indeed, the analyses and optimizations in these compilers work on more abstract representations, where programs are represented with data and control dependencies. The most representative representation is the sea-of-nodes form, used in the Java Hotspot Server Compiler, and which is the rationale behind the highly relaxed definition of the Java memory model. DISCOVER proposes to tackle the problem of verified compilation for shared-memory concurrency with a resolute language-based approach, and to investigate the formalization of adequate program intermediate representations and associated correctness proof techniques.

The project runs from October 2014 to September 2018.

5.3. European Initiatives

5.3.1. Collaborations in European Programs, Except FP7 & H2020

Program:CA COST Action CA15123

Project acronym: EUTYPES

Project title: European research network on types for programming and verification

Duration: 03/2016 to 03/2020

Coordinator: Herman Geuvers (Radboud University Nijmegen, The Netherlands)

Other partners: Austria, Belgium, Czech Republic, Denmark, Estonia, Finland, France, Macedonia, Germany, Hungary, Israel, Italy, Lithuania, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovenia, Spain, Sweden, United Kingdom

Abstract: Types are pervasive in programming and information technology. A type defines a formal interface between software components, allowing the automatic verification of their connections, and greatly enhancing the robustness and reliability of computations and communications. In rich dependent type theories, the full functional specification of a program can be expressed as a type. Type systems have rapidly evolved over the past years, becoming more sophisticated, capturing new aspects of the behaviour of programs and the dynamics of their execution.

This COST Action will give a strong impetus to research on type theory and its many applications in computer science, by promoting (1) the synergy between theoretical computer scientists, logicians and mathematicians to develop new foundations for type theory, for example as based on the recent development of "homotopy type theory", (2) the joint development of type theoretic tools as proof assistants and integrated programming environments, (3) the study of dependent types for programming and its deployment in software development, (4) the study of dependent types for verification and its deployment in software analysis and verification. The action will also tie together these different areas and promote cross-fertilisation.

5.4. International Initiatives

5.4.1. Inria Associate Teams Not Involved in an Inria International Labs

5.4.1.1. JCERT

Title: Verified Compilation of Concurrent Managed Languages

International Partner (Institution - Laboratory - Researcher):

Purdue University (United States) - School of Electrical and Computer Engineering (ECE) - Jan Vitek

Start year: 2014

See also: http://www.irisa.fr/celtique/ea/jcert/

Safety-critical applications demand rigorous, unambiguous guarantees on program correctness. While a combination of testing and manual inspection is typically used for this purpose, bugs latent in other components of the software stack, especially the compiler and the runtime system, can invalidate these hard-won guarantees. To address such concerns, additional laborious techniques such as manual code reviews of generated assembly code are required by certification agencies. Significant restrictions are imposed on compiler optimizations that can be performed, and the scope of runtime and operating system services that can be utilized. To alleviate this burden, the JCert project is implementing a verified compiler and runtime for managed concurrent languages like Java or C#.

5.4.2. Inria International Partners

5.4.2.1. WEBCERT

Title: Verified Trustworthy web Applications

International Partner (Institution - Laboratory - Researcher):

Imperial College (United Kingdom) - Department of Computing - Philippa Gardner

Duration: 2015 - 2019

Start year: 2015

See also: JSCert web page

The goal of the WebCert partnership is to extend the development of the JSCert formal semantics of JavaScript in the following domains: further mechanized specification, human-readable formal specification, program logic, verification tools, and the formalization of Defensive JavaScript.

5.4.2.2. Informal International Partners

Alan Schmitt is part of a Polonium Hubert Curien Partnership (PHC) with the University of Wrocław. This partnership is led by Sergueï Lenglet, from Loria, Nancy (currently visiting member of the Celtique project).

5.5. International Research Visitors

5.5.1. Visits of International Scientists

5.5.1.1. Internships

Thomas Wood

Date: Oct 2016 - Dec 2016

Institution: Imperial College (United Kingdom)

Ahmad Salim Al-Sibahi

Date: Sep 2016 - Jan 2017

Institution: IT University of Copenhagen (Denmark)

6. Dissemination

6.1. Promoting Scientific Activities

6.1.1. Scientific Events Organisation

6.1.1.1. General Chair, Scientific Chair

- PLMW@SPLASH 2016 (Programming Languages Mentoring Workshop) was chaired by Sandrine Blazy and Ulrik Prag-Schultz
- CoqPL 2017 (International Workshop on Coq for PL) was chaired by Sandrine Blazy and Emilio Jesus Gallego Arias

6.1.1.2. Member of the Organizing Committees

- JFLA 2016 (Journées Francophones des Langages Applicatifs) was locally organized by Julien Signoles and Alan Schmitt

6.1.2. Scientific Events Selection

6.1.2.1. Chair of Conference Program Committees

- VSTTE 2016 (Verified Software: Theories, Tools, and Experiments) was chaired by Sandrine Blazy and Marsha Chechik

6.1.2.2. Member of the Conference Program Committees

- CoqPL 2017 (International Workshop on Coq for PL) : Sandrine Blazy
- CPP 2017 (ACM SIGPLAN Conference on Certified Programs and Proofs) : Delphine Demange
- POPL 2017 (Symposium on Principles of Programming Languages) : Delphine Demange (External Program Committee)
- ESOP 2017 (European Symposium on Programming) : David Pichardie
- CC 2017 (International Conference on Compiler Construction) : David Pichardie
- IFL 2016 (International symposium on Implementation and application of Functional Languages) : Sandrine Blazy
- APLAS 2016 (Asian Symposium on Programming Languages and Systems) : Sandrine Blazy
- VSTTE 2016 (Verified Software: Theories, Tools, and Experiments) : Sandrine Blazy, Frédéric Besson
- GPCE 2016 (Generative Programming: Concepts & Experiences) : Sandrine Blazy
- DS@STAF 2016 (Doctoral Symposium) : Sandrine Blazy
- CPP 2016 (Certified Proofs and Programs) : Sandrine Blazy
- HaTT 2016 (International Workshop - Hammers for Type Theories) : Frédéric Besson
- AFADL 2016 (Approches Formelles dans l'Assistance au Développement de Logiciels) : Sandrine Blazy
- iFM 2016 (International Conference on integrated Formal Methods) : Delphine Demange
- FTfJP 2016 (Workshop on Formal Techniques for Java-like Programs) : Delphine Demange
- IFIP SEC 2016 (31st International Conference on ICT Systems Security and Privacy) : Thomas Jensen

6.1.2.3. Reviewer

- POPL 2017 (Symposium on Principles of Programming Languages): Alan Schmitt
- ESOP 2017 (European Symposium on Programming): Alan Schmitt
- VMCAI 2017 (International Conference on Verification, Model Checking, and Abstract Interpretation) : Delphine Demange

6.1.3. Journal

6.1.3.1. Reviewer - Reviewing Activities

- Journal of Software Evolution and Process: Sandrine Blazy
- International Journal of Computer Mathematics: Alan Schmitt
- Science of Computer Programming: Alan Schmitt

6.1.4. Invited Talks

- Journées nationales 2016 GDR Informatique Mathématique : Delphine Demange

6.1.5. Leadership within the Scientific Community

- Thomas Jensen is director of the Department NUMERIC of informatics, mathematics and electrical engineering at University Bretagne Loire.
- Thomas Jensen is leader of the security track of the LABEX Comin Labs.

6.1.6. Scientific Expertise

- Sandrine Blazy: expertise of 1 ANR project.
- Thomas Jensen: expertise of full project proposals for the ANR.

6.1.7. Research Administration

- Sandrine Blazy is member of Section 6 of the national committee for scientific research CoNRS from Sept. 2016.
- Sandrine Blazy is coordinator of the LTP (Languages, Types, Proofs) group of the French GDR GPL.

6.2. Teaching - Supervision - Juries

6.2.1. Teaching

Licence : Sandrine Blazy, Functional programming, 30h, L3, Université Rennes 1, France

Licence: Delphine Demange, Software Engineering, 40h, L2, Université de Rennes 1, France

Licence: Delphine Demange, Functional Programming, 75h, L1, Université de Rennes 1, France

Licence: Thomas Genet, Software Engineering, 58h, L2, Université de Rennes 1 / Istic, France

Licence : Alan Schmitt, Programmation Fonctionnelle, 72h (2 semestres), L3, Insa Rennes, France

Licence : David Pichardie, Algorithms, 36h, L3, ENS Rennes, France

Licence : David Cachera, Logic, 36h, L3, ENS Rennes, France

Master : Sandrine Blazy, Méthodes Formelles pour le développement de logiciels sûrs, 53h, M1, Université Rennes 1, France

Master : Thomas Genet, Formal Design and Verification, 108h, M1, Université de Rennes 1 / Istic, France

Master : Thomas Genet, Cryptographic Protocols, 24h, M2, Université de Rennes 1 / Istic, France

Master : David Pichardie, Mechanized Semantics, 15h, M2, Université Rennes 1, France

Master : Sandrine Blazy, Mechanized Semantics, 15h, M2, Université Rennes 1, France

Master : Sandrine Blazy, Semantics, 24h, M1, Université Rennes 1, France

Master : David Cachera, Semantics, 24h, M1, Université Rennes 1, France

Master : Sandrine Blazy, Software vulnerabilities, 20h, M2, Université Rennes 1, France

Master : Delphine Demange, Software Security, 9h, M2, Université Rennes 1, France

Master : Thomas Jensen, Program analysis and Software Security, 36h, M2, Université Rennes 1, France.

6.2.2. Supervision

PhD in progress : Alexandre Dang, Compiler for security, Octobre 2016, Thomas Jensen and Frédéric Besson

PhD in progress : Julien Lepiller, Binary Validation of Software Fault Isolation, Octobre 2016, Thomas Jensen and Frédéric Besson

PhD in progress : Gurvan Cabon, Analyse non locale certifiée en JavaScript grâce à une sémantique annotée, 1st september 2015, Alan Schmitt

PhD in progress : Florent Saudel, Vulnerability discovery, November 2015, Sandrine Blazy, Frédéric Besson and Dimitri Kirchner (Amossys)

PhD in progress : Alix Trieu, Formally verified compilation and static analysis, January 2016, Sandrine Blazy and David Pichardie

PhD in progress: David Bühler, Communication between analyses by deductive verification and abstract interpretation, November 2013, Sandrine Blazy and Boris Yakobowski (CEA)

PhD in progress : Yon Fernandez De Retana, Verified Optimising Compiler for high-level languages, 1st september 2015, David Pichardie and Delphine Demange

PhD in progress : Yannick Zakowski, Programs Logics for Concurrency, 1st september 2014, David Pichardie and David Cachera

PhD in progress : Oana Andreescu, Static analysis of functional specifications, 1st September 2013, Thomas Jensen, Stéphane Lescuyer (Prove & Run)

PhD in progress: Pauline Bolignano, Modeling and abstraction of system software, 1st November 2013, Thomas Jensen, Vincent Silés (Prove & Run)

Pierre Wilke, Formally verified compilation of low-level C code, Sandrine Blazy and Frédéric Besson, defended Nov 2016

Martin Bodin, Certified Analyses of JavaScript, Thomas Jensen and Alan Schmitt, defended Nov 2016

6.2.3. Juries

Sandrine Blazy, jury member (reviewer) for the PhD defense of Stefania Dumbrava, December 2016, Paris-Sud University, France

Sandrine Blazy, jury member (reviewer) for the PhD defense of Léon Gondelman, December 2016, Paris-Sud University, France

Sandrine Blazy, jury member (president) for the PhD defense of Thomas Degueule, December 2016, Rennes 1 University, France

Sandrine Blazy, jury member (president) for the PhD defense of Arjun Suresh, May 2016, Rennes 1 University, France

Sandrine Blazy, jury member for the selection of Inria CR (researcher) candidates, March and April 2016, Inria, Saclay, France.

Sandrine Blazy, jury member for the selection of a professeur at University of Perpignan, May 2016, Perpignan, France.

Alan Schmitt, jury member for the selection of Inria CR (researcher) candidates, March and April 2016, Inria, Rennes, France.

Delphine Demange, jury member for the selection of a Maître de Conférences at University Paris Diderot (Paris 7) / IRIF, May 2016, Paris, France.

Alan Schmitt, jury member (reviewer) for the PhD defense of Régis Spadotti, May 2016, Université Toulouse III

Alan Schmitt, jury member (reviewer) for the HDR defense of Nicolas Tabareau, November 2016, Université de Nantes

Thomas Jensen, jury member (reviewer) for the PhD defense of Denis Martinez, February 2016, Université de Montpellier

Thomas Jensen, jury member for the PhD defense of Oliver Schwarz, October 2016, KTH, Stockholm, Sweden

Thomas Jensen, jury member (reviewer) for the PhD defense of Rabah Laouadi, December 2016, Université de Montpellier

David Pichardie, jury member for the PhD defense of Jacques-Henri Jourdan, May 2016, Université de Paris Diderot

6.3. Popularization

Talk “Bug, Virus, Intrusion, Pirates... So many threats and no defense? Yes... maths.”, Thomas Genet, given three times in high schools close to Rennes.

7. Bibliography

Major publications by the team in recent years

[1] G. BARTHE, D. DEMANGE, D. PICHARDIE. *Formal Verification of an SSA-based Middle-end for CompCert*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2014, 35 p. , https://hal.inria.fr/hal-01097677

[2] F. BESSON, N. BIELOVA, T. JENSEN. *Hybrid Information Flow Monitoring Against Web Tracking*, in "CSF - 2013 IEEE 26th Computer Security Foundations Symposium", New Orleans, United States, 2013 [*DOI :* 10.1109/CSF.2013.23], http://hal.inria.fr/hal-00924138

[3] F. BESSON, T. JENSEN, D. PICHARDIE. *Proof-Carrying Code from Certified Abstract Interpretation to Fixpoint Compression*, in "Theoretical Computer Science", 2006, vol. 364, n^{o} 3, pp. 273–291

[4] M. BODIN, A. CHARGUÉRAUD, D. FILARETTI, P. GARDNER, S. MAFFEIS, D. NAUDZIUNIENE, A. SCHMITT, G. SMITH. *A Trusted Mechanised JavaScript Specification*, in "POPL 2014 - 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", San Diego, United States, November 2013, http://hal.inria.fr/hal-00910135

[5] B. BOYER, T. GENET, T. JENSEN. *Certifying a Tree Automata Completion Checker*, in "4th International Joint Conference, IJCAR 2008", Lectures Notes in Computer Science, Springer-Verlag, 2008, vol. 5195, pp. 347–362

[6] D. CACHERA, T. JENSEN, A. JOBIN, F. KIRCHNER. *Inference of polynomial invariants for imperative programs: a farewell to Gro¨bner bases*, in "Science of Computer Programming", 2014, vol. 93, 21 p. [*DOI :* 10.1016/J.SCICO.2014.02.028], https://hal.inria.fr/hal-00932351

[7] D. CACHERA, T. JENSEN, D. PICHARDIE, V. RUSU. *Extracting a Data Flow Analyser in Constructive Logic*, in "Theoretical Computer Science", 2005, vol. 342, n^{o} 1, pp. 56–78

[8] D. DEMANGE, V. LAPORTE, L. ZHAO, D. PICHARDIE, S. JAGANNATHAN, J. VITEK. *Plan B: A Buffered Memory Model for Java*, in "Proc. of the 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013", Rome, Italy, ACM, 2013, http://hal.inria.fr/hal-00924716

[9] T. GENET, V. RUSU. *Equational Approximations for Tree Automata Completion*, in "Journal of Symbolic Computation", 2010, vol. 45(5):574-597, May 2010, n^{o} 5, pp. 574-597

[10] L. HUBERT, T. JENSEN, V. MONFORT, D. PICHARDIE. *Enforcing Secure Object Initialization in Java*, in "15th European Symposium on Research in Computer Security (ESORICS)", Lecture Notes in Computer Science, Springer, 2010, vol. 6345, pp. 101-115

[11] J.-H. JOURDAN, V. LAPORTE, S. BLAZY, X. LEROY, D. PICHARDIE. *A formally-verified C static analyzer*, in "POPL 2015: 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", Mumbai, India, ACM, January 2015, pp. 247-259 [*DOI :* 10.1145/2676726.2676966], https://hal.inria.fr/hal-01078386

Publications of the year

Articles in International Peer-Reviewed Journals

[12] A. AZEVEDO DE AMORIM, N. COLLINS, A. DEHON, D. DEMANGE, C. HRIŢCU, D. PICHARDIE, B. C. PIERCE, R. POLLACK, A. TOLMACH. *A Verified Information-Flow Architecture*, in "Journal of Computer Security (JCS); Special Issue on Verified Information Flow Security", December 2016, vol. 24, n^{o} 6, pp. 689–734, https://hal.archives-ouvertes.fr/hal-01424797

[13] N. BIELOVA, F. BESSON, T. JENSEN. *Using JavaScript Monitoring to Prevent Device Fingerprinting*, in "ERCIM News", July 2016, https://hal.inria.fr/hal-01353997

[14] S. BLAZY, D. BÜHLER, B. YAKOBOWSKI. *Improving static analyses of C programs with conditional predicates*, in "Science of Computer Programming", March 2016, vol. 118, Extended version of the FMICS 2014 paper [*DOI :* 10.1145/2854065.2854082], https://hal.inria.fr/hal-01242077

[15] S. BLAZY, V. LAPORTE, D. PICHARDIE. *Verified Abstract Interpretation Techniques for Disassembling Low-level Self-modifying Code*, in "Journal of Automated Reasoning", 2016, vol. 56, n^{o} 3, 26 p. , Version étendue de l'article de la conférence ITP 2014 [*DOI :* 10.1007/S10817-015-9359-8], https://hal.inria.fr/hal-01243700

[16] T. GENET. *Termination criteria for tree automata completion*, in "Journal of Logic and Algebraic Methods in Programming", 2016, vol. 85, Issue 1, part 1, pp. 3-33 [*DOI :* 10.1016/J.JLAMP.2015.05.003], https://hal.inria.fr/hal-01194533

[17] F. HONSELL, L. LIQUORI, P. MAKSIMOVIC, I. SCAGNETTO. *LLFP : A Logical Framework for modeling External Evidence, Side Conditions, and Proof Irrelevance using Monads*, in "Logical Methods in Computer Science", February 2016, https://hal.inria.fr/hal-01146059

International Conferences with Proceedings

[18] F. BESSON, N. BIELOVA, T. JENSEN. *Hybrid Monitoring of Attacker Knowledge*, in "29th IEEE Computer Security Foundations Symposium", Lisboa, Portugal, 2016, https://hal.inria.fr/hal-01310572

[19] S. BLAZY, V. LAPORTE, D. PICHARDIE. *An Abstract Memory Functor for Verified C Static Analyzers*, in "ACM SIGPLAN International Conference on Functional Programming (ICFP 2016)", Nara, Japan, ACM, September 2016, 14 p. [*DOI :* 10.1145/2951913.2951937], https://hal.inria.fr/hal-01339969

[20] S. BLAZY, A. TRIEU. *Formal Verification of Control-flow Graph Flattening*, in "Certified Proofs and Programs (CPP 2016)", Saint-Petersburg, United States, ACM (editor), Certified Proofs and Programs (CPP 2016), January 2016, 12 p. , forthcoming [*DOI :* 10.1145/2854065.2854082], https://hal.inria.fr/hal-01242063

[21] M. BODIN, T. JENSEN, A. SCHMITT. *An Abstract Separation Logic for Interlinked Extensible Records*, in "Vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016)", Saint-Malo, France, J. SIGNOLES (editor), January 2016, https://hal.archives-ouvertes.fr/hal-01333600

[22] P. BOLIGNANO, T. JENSEN, V. SILES. *Modeling and Abstraction of Memory Management in a Hypervisor*, in "Fundamental Approaches to Software Engineering (FASE'16)", Eindhoven, Netherlands, Proc. of Fundamental Approaches to Software Engineering (FASE'16), Springer, April 2016, vol. 9633, pp. 214 - 230 [*DOI :* 10.1007/978-3-662-49665-7_13], https://hal.inria.fr/hal-01394174

[23] G. CABON, D. CACHERA, D. PICHARDIE. *An Extended Buffered Memory Model With Full Reorderings*, in "FtFjp - Ecoop workshop", Rome, Italy, July 2016, pp. 1 - 6 [*DOI :* 10.1145/2955811.2955816], https://hal.inria.fr/hal-01379514

[24] D. DEMANGE, Y. FERNANDEZ DE RETANA. *Mechanizing conventional SSA for a verified destruction with coalescing*, in "25th International Conference on Compiler Construction", Barcelona, Spain, March 2016, https://hal.archives-ouvertes.fr/hal-01378393

[25] C. FOURNET, C. KELLER, V. LAPORTE. *A Certified Compiler for Verifiable Computing*, in "IEEE 29th Computer Security Foundations Symposium, CSF 2016", Lisbonne, Portugal, June 2016, https://hal.inria.fr/hal-01397680

[26] D. KÄSTNER, X. LEROY, S. BLAZY, B. SCHOMMER, M. SCHMIDT, C. FERDINAND. *Closing the Gap – The Formally Verified Optimizing Compiler CompCert*, in "SSS'17: Safety-critical Systems Symposium 2017", Bristol, United Kingdom, Proceedings of the Twenty-fifth Safety-Critical Systems Symposium, February 2017, https://hal.inria.fr/hal-01399482

[27] A. OANA, T. JENSEN, S. LESCUYER. *Correlating Structured Inputs and Outputs in Functional Specifications*, in "Software Engineering and Formal Methods", Vienna, Austria, 14th Int. Software Engineering and Formal Methods conference, Springer, July 2016, vol. 9763, 19 p. [*DOI :* 10.1007/978-3-319-41591-8_7], https://hal.inria.fr/hal-01394178

Conferences without Proceedings

[28] X. LEROY, S. BLAZY, D. KÄSTNER, B. SCHOMMER, M. PISTER, C. FERDINAND. *CompCert - A Formally Verified Optimizing Compiler*, in "ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress", Toulouse, France, SEE, January 2016, https://hal.inria.fr/hal-01238879

Books or Proceedings Editing

[29] S. BLAZY, M. CHÉCHIA (editors). *Verified Software: Theories, Tools, and Experiments - 8th International Conference, VSTTE 2016, Toronto, Canada, July 17-18, 2016. Proceedings*, Lecture Notes in Computer Science, Springer, Toronto, Canada, 2016, vol. 9971, https://hal.inria.fr/hal-01387207

References in notes

[30] G. BARTHE, D. DEMANGE, D. PICHARDIE. *Formal Verification of an SSA-based Middle-end for CompCert*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2014, 35 p. , https://hal.inria.fr/hal-01097677

[31] S. BLAZY, D. DEMANGE, D. PICHARDIE. *Validating Dominator Trees for a Fast, Verified Dominance Test*, in "Proc. of the 6th International Conference on Interactive Theorem Proving (ITP 2015)", LNCS, Springer, 2015

[32] M. BODIN, T. JENSEN, A. SCHMITT. *Certified Abstract Interpretation with Pretty-Big-Step Semantics*, in "Certified Programs and Proofs (CPP 2015)", Mumbai, India, Proceedings of the 2015 Conference on Certified Programs and Proofs, January 2015 [*DOI :* 10.1145/2676724.2693174], https://hal.inria.fr/hal-01111588

[33] B. BOISSINOT, A. DARTE, F. RASTELLO, B. DUPONT DE DINECHIN, C. GUILLON. *Revisiting Out-of-SSA Translation for Correctness, Code Quality and Efficiency*, in "Proc. of CGO'09", IEEE Computer Society, 2009, pp. 114–125

[34] G. BOUDOL, G. PETRI, B. P. SERPETTE. *Relaxed Operational Semantics of Concurrent Programming Languages*, in "EXPRESS/SOS", 2012, vol. 89, pp. 19-33

[35] D. DEMANGE, V. LAPORTE, L. ZHAO, D. PICHARDIE, S. JAGANNATHAN, J. VITEK. *Plan B: A Buffered Memory Model for Java*, in "Proc. of the 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013", Rome, Italy, ACM, 2013, http://hal.inria.fr/hal-00924716

[36] D. DEMANGE, L. STEFANESCO, D. PICHARDIE. *Verifying Fast and Sparse SSA-based Optimizations in Coq*, in "Proc. of CC'15", LNCS, 2015, vol. 9031, pp. 233-252

[37] J.-H. JOURDAN, V. LAPORTE, S. BLAZY, X. LEROY, D. PICHARDIE. *A formally-verified C static analyzer*, in "POPL 2015: 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", Mumbai, India, ACM, January 2015, pp. 247-259 [*DOI :* 10.1145/2676726.2676966], https://hal.inria.fr/hal-01078386

[38] J. A. KROLL, G. STEWART, A. W. APPEL. *Portable Software Fault Isolation*, in "CSF 2014", IEEE, 2014, pp. 18–32, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6954678

[39] P. WILKE. *Formally verified compilation of low-level C code*, University of Rennes 1, 2016