Activity Report 2018

Team CELTIQUE

Software Certification with Semantic Analysis

Joint team with Inria Rennes – Bretagne Atlantique

D4 – Language and Software Engineering

Table of contents

Project-Team CELTIQUE

Creation of the Project-Team: 2009 July 01

Keywords:

Computer Science and Digital Science:

A2.1. - Programming Languages
A2.1.1. - Semantics of programming languages
A2.1.3. - Object-oriented programming
A2.1.4. - Functional programming
A2.1.12. - Dynamic languages
A2.2. - Compilation
A2.2.1. - Static analysis
A2.2.2. - Memory models
A2.2.3. - Memory management
A2.2.5. - Run-time systems
A2.2.9. - Security by compilation
A2.4. - Formal method for verification, reliability, certification
A2.4.1. - Analysis
A2.4.2. - Model-checking
A2.4.3. - Proofs
A4. - Security and privacy
A4.5. - Formal methods for security
A7.2.2. - Automated Theorem Proving
A7.2.3. - Interactive Theorem Proving

Other Research Topics and Application Domains:

B6.1. - Software industry
B6.1.1. - Software engineering
B6.4. - Internet of things
B6.6. - Embedded systems
B9.10. - Privacy

1. Team, Visitors, External Collaborators

Research Scientists

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

Faculty Members

Sandrine Blazy [Univ de Rennes I, Professor, HDR]
David Cachera [Ecole normale supérieure de Rennes, Associate Professor, HDR]
Delphine Demange [Univ de Rennes I, Associate Professor]
Thomas Genet [Univ de Rennes I, Associate Professor, HDR]
Serguei Lenglet [Univ de Lorraine, Associate Professor, until Aug 2018]

David Pichardie [Ecole normale supérieure de Rennes, Professor, HDR]

Post-Doctoral Fellows

Stefania Dumbrava [Ecole normale supérieure de Rennes, from Sep 2018]
Jean-Christophe Léchenet [Ecole normale supérieure de Rennes, from Sep 2018]
Thomas Rubiano [Univ de Rennes I, from Sep 2018]

PhD Students

Gurvan Cabon [Inria, until Nov 2018]
Alexandre Dang [Inria]
Samy Daoud [Orange Labs, from Nov 2018]
Yon Fernandez de Retana [Univ de Rennes I, until Aug 2018]
Timothée Haudebourg [Univ de Rennes I]
Rémi Hutin [Ecole normale supérieure de Rennes, from Sep 2018]
Julien Lepiller [Inria]
Florent Saudel [AMOSSYS, until Apr 2018]
Alix Trieu [Univ de Rennes I]

Technical staff

Nicolas Barré [Ecole normale supérieure de Rennes, from Sep 2018]
Samuel Risbourg [Inria, from Sep 2018]
Yannick Zakowski [Univ de Rennes I, until Jan 2018]

Interns

Guillaume Ambal [Ecole Normale Supérieure Lyon, until Jun 2018]
Ronan Buron [Univ de Rennes I, from Apr 2018 until Jun 2018]
Nathanael Courant [Ecole Normale Supérieure Paris, from Sep 2018]
Enzo Crance [Inria, from Jun 2018 until Aug 2018]
Rémi Hutin [Ecole normale supérieure de Rennes, from Feb 2018 until Jun 2018]
Vaikunt Mallya [Ecole normale supérieure de Rennes, from May 2018 until Aug 2018]

Administrative Assistant

Lydie Mabil [Inria]

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. Highlights of the Year

3.1. Highlights of the Year

3.1.1. Awards

- The ERC Consolidator grant VESTA on verified static analysis was awarded to David Pichardie and launched in September 2018.
- The ANR project SCRYPT led by Frédéric Besson was accepted and starts in February 2019.

BEST PAPER AWARD:

[19]
A. SALIM AL-SIBAHI, A. S. DIMOVSKI, T. JENSEN, A. WASOWSKI. *Verification of High-Level Transformations with Inductive Refinement Types*, in "GPCE 2018 - 17th International Conference on Generative Programming: Concepts & Experience", Boston, United States, November 2018, pp. 1-14, https://hal.inria.fr/hal-01898058

4. New Software and Platforms

4.1. CompcertSSA

KEYWORDS: Optimizing compiler - Formal methods - Proof assistant - SSA
FUNCTIONAL DESCRIPTION: CompcertSSA is built on top of the Compcert verified C compiler, by adding a middle-end based on the SSA form (Static Single Assignment) : conversion to SSA, SSA-based optimizations, and destruction of SSA.

- Participants: Sandrine Blazy, Delphine Demange, Yon Fernandez De Retana, David Pichardie and Leo Stefanesco
- Contact: Delphine Demange
- Publications: Mechanizing conventional SSA for a verified destruction with coalescing - Validating Dominator Trees for a Fast, Verified Dominance Test - A Formally Verified SSA-based Middle-end - Formal Verification of an SSA-based Middle-end for CompCert - Verifying Fast and Sparse SSA-based Optimizations in Coq.
- URL: http://compcertssa.gforge.inria.fr/

4.2. Jacal

JAvaCard AnaLyseur
KEYWORDS: JavaCard - Certification - Static program analysis - AFSCM
FUNCTIONAL DESCRIPTION: Jacal is a JAvaCard AnaLyseur developed on top of the SAWJA platform. This proprietary software verifies automatically that Javacard programs conform with the security guidelines issued by the AFSCM (Association Française du Sans Contact Mobile). Jacal is based on the theory of abstract interpretation and combines several object-oriented and numeric analyses to automatically infer sophisticated invariants about the program behaviour. The result of the analysis is thereafter harvest to check that it is sufficient to ensure the desired security properties.

- Participants: David Pichardie, Delphine Demange, Frédéric Besson and Thomas Jensen
- Contact: Thomas Jensen

4.3. Javalib

KEYWORDS: Library - Java - Ocaml
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: David Pichardie, Frédéric Besson, Laurent Guillo, Laurent Hubert, Nicolas Barré, Pierre Vittet and Tiphaine Turpin
- Contact: David Pichardie
- URL: http://sawja.inria.fr/

4.4. JSCert

Certified JavaScript
FUNCTIONAL DESCRIPTION: The JSCert project aims to really understand JavaScript. 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: Alan Schmitt and Martin Bodin
- Partner: Imperial College London
- Contact: Alan Schmitt
- URL: http://jscert.org/

4.5. 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: David Pichardie, Frédéric Besson and Laurent Guillo
- Partners: CNRS - ENS Cachan
- Contact: David Pichardie
- URL: http://sawja.inria.fr/

4.6. Timbuk

KEYWORDS: Automated deduction - Ocaml - Program verification - Tree Automata - Term Rewriting Systems
FUNCTIONAL DESCRIPTION: Timbuk is a tool designed to compute or over-approximate sets of terms reachable by a given term rewriting system. The libray also provides an OCaml toplevel with all usual functions on Bottom-up Nondeterministic Tree Automata.

- Participant: Thomas Genet
- Contact: Thomas Genet
- URL: http://people.irisa.fr/Thomas.Genet/timbuk/index.html

4.7. jsexplain

JSExplain
KEYWORDS: JavaScript - Compilation - Standards - Debug - Interpreter
FUNCTIONAL DESCRIPTION: JSExplain is a reference interpreter for JavaScript that closely follows the specification and that produces execution traces. These traces may be interactively investigated in a browser, with an interface that displays not only the code and the state of the interpreter, but also the code and the state of the interpreted program. Conditional breakpoints may be expressed with respect to both the interpreter and the interpreted program. In that respect, JSExplain is a double-debugger for the specification of JavaScript.

- Partner: Imperial College London
- Contact: Alan Schmitt
- Publication: JSExplain: A Double Debugger for JavaScript
- URL: https://github.com/jscert/jsexplain

5. New Results

5.1. Software Fault Isolation

Participants: Frédéric Besson, Thomas Jensen, Julien Lepiller.

Software Fault Isolation (SFI) consists in transforming untrusted code so that it runs within a specific address space, (called the sandbox) and verifying at load-time that the binary code does indeed stay inside the sandbox. Security is guaranteed solely by the SFI verifier whose correctness therefore becomes crucial. Existing verifiers enforce a very rigid, almost syntactic policy where every memory access and every control-flow transfer must be preceded by a sandboxing instruction sequence, and where calls outside the sandbox must implement a sophisticated protocol based on a shadow stack. We have defined SFI as a defensive semantics, with the purpose of deriving semantically sound verifiers that admit flexible and efficient implementations of SFI. We derive an executable analyser, that works on a per-function basis, which ensures that the defensive semantics does not go wrong, and hence that the code is well isolated. Experiments show that our analyser exhibits the desired flexibility: it validates correctly sandboxed code, it catches code breaking the SFI policy, and it can validate programs where redundant instrumentations are optimised away [8].

5.2. Compilation and Side-Channels

Participants: Frédéric Besson, Alexandre Dang, Thomas Jensen.

The usual guarantee provided by compilers is that the input/output observable behaviour of the target program is one of the possible behaviours of the source program. In the context of security, the notion of observable behaviour needs to be revisited in order to take into account side-channels *i.e.* observations beyond input/output that are leaked by the program execution to a potential attacker.

For instance, a common security recommendation is to reduce the in-memory lifetime of secret values, in order to reduce the risk that an attacker can obtain secret data by probing memory. To mitigate this risk, secret values can be overwritten, at source level, after their last use. However, as secret values are never used afterwards, a compiler may remove this mitigation during a standard Dead Store Elimination pass. We propose a formal definition of Information Flow Preserving transformation [7] which ensures that secret values are not easier to obtain at assembly level than at source level. Using the notion of Attacker Knowledge, we relate the information leak of a program before and after the transformation. We consider two classic compiler passes (Dead Store Elimination and Register Allocation) and show how to validate and, if needed, modify these transformations in order to be information flow preserving.

5.3. Semantic study of the Sea-of-Nodes form

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

As part of the PhD of Yon Fernandez de Retana [2], we started to study the the Sea-of-Nodes form. This intermediate representation was introduced by Cliff Click in the mid 90s [21] as an enhanced SSA form. It improves on the initial SSA form by relaxing the total order on instructions in basic blocks into explicit data and control dependencies. This makes programs more flexible to optimize. While Sea-of-node is popular in many production-size compiler (Sun's HotSpot, Graal...), it is still not very well understood, from a semantic, foundational point of view. We have defined a simple but rigorous formal semantics for a Sea-of-Nodes form. It comprises a denotational component to express data computation, and an operational component to express control flow. We prove a fundamental, dominance-based semantic property on Sea-of-Nodes programs which determines the regions of the graph where the values of nodes are preserved. Finally, we apply our results to prove the semantic correctness of a redundant zero-check elimination optimization. All the necessary semantic properties have been mechanically verified in the Coq proof assistant. These results have been published in [11]. A more detailed account can be found in Yon Fernandez de Retana's PhD manuscript [2].

5.4. Certified Concurrent Garbage Collector

Participants: David Cachera, Delphine Demange, David Pichardie, Yannick Zakowski.

Concurrent garbage collection algorithms are an emblematic challenge in the area of concurrent program verification. We addressed this problem by proposing a mechanized proof methodology based on the popular Rely-Guarantee (RG) proof technique. We designed a specific compiler intermediate representation (IR) with strong type guarantees, dedicated support for abstract concurrent data structures, and high-level iterators on runtime internals (objects, roots, fields, thread identifiers...). In addition, we defined an RG program logic supporting an incremental proof methodology where annotations and invariants can be progressively enriched. We have formalized the IR, the proof system, and proved the soundness of the methodology in the Coq proof assistant. Equipped with this IR, we have proved the correctness of a fully concurrent garbage collector where mutators never have to wait for the collector. This work has been published in [6] as an extended version of [22].

In this work, reasoning simultaneously about the garbage collection algorithm and the concrete implementation of the concurrent data-structures it uses would have entailed an undesired and unnecessary complexity. The above proof is therefore conducted with respect to abstract operations which execute atomically. In practice, however, concurrent data-structures uses fine-grained concurrency, for performance reasons. One must therefore prove an observational refinement between the abstract concurrent data-structures and their fined-grained, “linearisable” implementation. To adress this issue, we introduce a methodology inspired by the work of Vafeiadis, and provide the approach with solid semantic foundations. Assuming that fine-grained implementations are proved correct with respect to an RG specification encompassing linearization conditions, we prove, once and for all, that this entails a semantic refinement of their abstraction. This methodology is instantiated to prove correct the main data-structure used in our garbage collector. This work has been published in [20].

5.5. Formalization of Higher-Order Process Calculi

Participants: Guillaume Ambal, Sergueï Lenglet, Alan Schmitt.

Guillaume Amabal, Sergueï Lenglet, Alan Schmitt have continued exploring how to formalize HOπ in Coq, in particular how to deal with the different kinds of binders used in the calculus. We have studied and compared several approaches, such as locally nameless, De Bruijn indices, and nominal binders. We have discovered that the locally nameless approach introduces a lot of complexity, as name restriction allows reduction under binders, introducing the need for numerous renaming lemmas. The nominal approach is quite elegant and very close to the pen and paper definitions, but it still requires many technical lemmas to be proven. The de Bruijn approach is the most concise. A first version of this work has been published at CPP 2018 [18] and a journal version is submitted for publication. The Coq scripts can be found at http://passivation.gforge.inria.fr/hopi/.

5.6. Certified Semantics and Analyses for JavaScript

Participants: Samuel Risbourg, Alan Schmitt.

Alan Schmitt has continued his collaboration with Arthur Charguéraud (Inria Nancy) and Thomas Wood (Imperial College London) to develop JSExplain, an interpreter for JavaScript that is as close as possible to the specification. Since September 2018, Samuel Risbourg has been hired to continue developing the tool. It is publicly available at https://github.com/jscert/jsexplain and is described in a publication at The Web Conference 2018 [10]. The tool is regularly presented at the TC39 committee standardizing JavaScript to solicit feedback.

5.7. Skeletal Semantics

Participants: Nathanael Courant, Thomas Jensen, Alan Schmitt.

Alan Schmitt and Thomas Jensen, in collaboration with Martin Bodin and Philippa Gardner at Imperial College London, have designed a new meta-language to formally describe semantics. A fundamental idea behind this approach to semantics is that this description can be used to derive several *interpretations* corresponding to different kinds of semantics, such as a big-step semantics, an abstract interpretation, or a control-flow analysis. The correctness of these semantics is proven independently of the language considered. This work has been accepted at POPL 2019 [4] and is formalized in Coq (see the skeletal semantics web site for more detail). Nathanael Courant is currently extending this work to generate analyses automatically and to facilitate the way in which to adjust their precision.

5.8. Verification of High-Level Transformations with Inductive Refinement Types

Participant: Thomas Jensen.

High-level transformation languages like Rascal or Stratego include expressive features for manipulating large abstract syntax trees: first-class traversals, expressive pattern matching, backtracking and generalized iterators. We have designed and implemented an abstract interpretation tool, Rabit, for verifying inductive type and shape properties for transformations written in such languages. We describe how to perform abstract interpretation based on operational semantics, specifically focusing on the challenges arising when analyzing the expressive traversals and pattern matching. We have evaluated Rabit on a series of transformations (normalization, desugaring, refactoring, code generators, type inference, etc.) showing that we can effectively verify stated properties.

This work was done in collaboration with researchers at the IT University of Copenhagen. The paper [19] presenting these results won the Best Paper award at GPCE 2018.

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

Participants: Thomas Genet, Thomas Jensen, Timothée Haudebourg.

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. We have already shown that tree automata completion can safely over-approximate the image of any first-order complete and terminating functional program. This year we successfully extended this result to the case of higher-order functional programs [15], [16]. Moreover, the approximation automaton can be certified using an efficient Coq-extracted checker that we developped in 2008. Thus, we have an automatic static analysis procedure for higher-order functional programs whose results are certified by the Coq proof assistant. The algorithm presented in [15] has been implemented in Timbuk [14] and gives very encouraging experimental results http://people.irisa.fr/Thomas.Genet/timbuk/funExperiments/. Besides, we have shown the completeness of this approach, i.e., that any regular approximation of the image of a function can be found using completion [13].

6. Partnerships and Cooperations

6.1. National Initiatives

6.1.1. 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.

6.1.2. 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.

6.1.3. The ANR AJACS project

Participants: 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 March 2019.

6.1.4. The ANR DISCOVER project

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

The DISCOVER project project (2014–09/2019) 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 2019.

6.1.5. The ANR CISC project

Participants: Frédéric Besson, Thomas Jensen, Alan Schmitt.

The goal of the CISC project is to investigate multitier languages and compilers to build secure IoT applications with private communication. In particular, we aim at extending multitier platforms by a new orchestration language that we call Hiphop.js to synchronize internal and external activities of IoT applications as a whole. Our goal is to define language, semantics, attacker models, and policies for the IoT and investigate automatic implementation of privacy and security policies by multitier compilation of IoT applications. To guarantee such applications are correct, and in particular that the required security and privacy properties are achieved, we propose to certify them using the Coq proof assistant. We plan to implement the CISC results as extensions of the multitier language Hop.js (developed at Inria), based on the JavaScript language to maximize its impact. Using the new platform, we will carry out experimental studies on IoT security.

The project partners include the following Inria teams: Celtique, Collège de France, Indes, and Privatics. The project runs from April 2018 to March 2022.

6.2. European Initiatives

6.2.1. FP7 & H2020 Projects

6.2.1.1. The ERC VESTA project

Participants: David Pichardie, Sandrine Blazy, Nicolas Barré, Stefania Dumbrava, Jean-Christophe Léchenet, Rémi Hutin.

The VESTA project aims at proposing guidance and tool-support to the designers of static analysis, in order to build advanced but reliable static analysis tools. We focus on analyzing low-level softwares written in C, leveraging on the CompCert verified compiler. Verasco is a verified static analyser that analyses C programs and follows many of the advanced abstract interpretation technique developped for Astrée. The outcome of the VESTA project will be a platform that help designing other verified advanced abstract interpreters like Verasco, without starting from a white page. We will apply this technique to develop security analyses for C programs. The platform will be open-source and will help the adoption of abstract interpretation techniques.

This a consolidator ERC awarded to David Pichardie for 5 year. The project started in september 2018.

6.2.2. 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.

Sandrine Blazy is Substitute Member of the Managment Committee for France.

6.3. International Initiatives

6.3.1. Inria International Partners

6.3.1.1. Declared Inria International Partners

WEBCERT

Title: Verified Trustworthy web Applications

International Partner (Institution - Laboratory - Researcher):

Imperial College London - Department of Computing - Philippa Gardner

Duration: 2015 - 2019

Start year: 2015

See also: JSCert web page

The WebCert partnership focuses on applying formal methods to the JavaScript language: mechanized specification, development of an executable formal specification, design of a program logic, development of verification tools, and study of secure sub-languages.

7. Dissemination

7.1. Promoting Scientific Activities

7.1.1. Scientific Events Organisation

7.1.1.1. Member of the Organizing Committees

- Formal Methods meet JavaScript Workshop 2018, organizer: Alan Schmitt

7.1.2. Scientific Events Selection

7.1.2.1. Chair of Conference Program Committees

- AVoCS 2018 (International workshop on automated verification of critical systems), FLOC workshop, Oxford (co-chair) : David Pichardie

7.1.2.2. Member of the Conference Program Committees

- CC 2018 (International Conference on Compiler Construction) : David Pichardie
- CPP 2018 (ACM SIGPLAN Conference on Certified Programs and Proofs) : Sandrine Blazy
- ITP 2018 (International Conference on Interactive Theorem Proving) : Delphine Demange
- Euro S&P 2018 (IEEE European Symposium on Security and Privacy) : Sandrine Blazy
- ICFP 2018 (ACM SIGPLAN International Conference on Functional Programming) : Sandrine Blazy
- LOPSTR 2018 (Symposium on Logic-Based Program Synthesis and Transformation) : Sandrine Blazy
- SAS 2018 artefact evaluation committee : Frédéric Besson
- ProWeb 2018: Alan Schmitt

7.1.3. Journal

7.1.3.1. Reviewer - Reviewing Activities

- Discrete Mathematics & Theoretical Computer: Alan Schmitt
- Theoretical Computer Science: Alan Schmitt
- Theoretical Informatics and Applications: Alan Schmitt

7.1.4. Invited Talks

- F. Besson gave a talk at the Dagstuhl Seminar 18201 "Secure Compilation".
- T. Genet gave an invited talk at the ETAPS “Workshop on Rewriting Logic and Applications” in Thessaloniki [12].

7.1.5. Leadership within the Scientific Community

- Sandrine Blazy coordinated of the LTP (Languages, Types, Proofs) group of the French GDR GPL (until October 2018).
- Thomas Jensen is head of the research axis “Security and Privacy” at the Labex CominLabs.
- Sandrine Blazy and Thomas Jensen are member of the Steering Committee of Static Analysis Symposium (SAS).

7.1.6. Scientific Expertise

- Sandrine Blazy: expertise of an ERC Advanced Grant research proposal
- David Pichardie: expertise of two ERC Advanced Grant research proposals
- Sandrine Blazy and David Pichardie : members of the HCERES evaluation committee of the LSV laboratory in December 2018
- David Pichardie: expertise of 1 ANR project
- David Pichardie, jury member for the selection of a Maître de Conférences at ENSEEIHT (Toulouse), May 2018
- Thomas Jensen: expertise of an ERC Consolidator Grant research proposal
- Alan Schmitt: expertise of a Innovational Research Incentives Scheme Veni research proposal

- Sandrine Blazy is member of IFIP WG 2.11 on program generation and of IFIP WG 1.9/2.15 on verified software

7.1.7. Research Administration

- Sandrine Blazy is member of Section 6 of the national committee for scientific research CoNRS.

7.2. Teaching - Supervision - Juries

7.2.1. Teaching

Licence : Sandrine Blazy, Programmation de confiance, 81h, L3, Université Rennes 1, France

Licence : David Cachera, Initiation to Pedagogy and Mediation, 11h, L3, ENS Rennes, France

Licence : David Cachera, Algorithmics, 20h, L3, ENS Rennes, France

Licence : Delphine Demange, Spécialité Informatique 1 - Algorithmique et Complexité Expérimentale, 36h, L1, Université Rennes 1, France

Licence : Delphine Demange, Spécialité Informatique 2 - Functional and Immutable Programming, 70h, L1, Université Rennes 1, France

Licence : Delphine Demange, Programmation de Confiance, 36h, L3, Université Rennes 1, France

Licence : Thomas Genet, Spécialité Informatique 1 - Algorithmique et Complexité Expérimentale, 47h, L1, Université Rennes 1, France

Licence : Thomas Genet, Initiation au génie logiciel, 67h, L2, Université Rennes 1, France

Licence : Thomas Jensen, Programmation de confiance, 20h, L3, Université Rennes 1, France

Licence : David Pichardie, Graph algorithms, 24h, L3, ESIR, France

Licence : Alan Schmitt, Programmation Fonctionnelle, 36h, L3, Insa Rennes, France

Licence : Alan Schmitt, Programmation de Confiance, 28h, L3, Université Rennes 1, France France

Master : Sandrine Blazy, Static analysis, 30h, M1, Université Rennes 1, France

Master : Sandrine Blazy, Mechanized Semantics, 30h, M2, Université Rennes 1, France

Master : Sandrine Blazy, Software vulnerabilities, 26h, M2, Université Rennes 1, France

Master : David Cachera, Semantics of Programming Languages, 28h, M1, Université Rennes 1, France

Master : David Cachera, Advanced Semantics, 20h, M2, ENS Rennes, France

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

Master : Thomas Genet, Analyse et conception formelle, 65h, M1, Université Rennes 1, France

Master : David Pichardie, Static analysis, 30h, M1, Université Rennes 1, France

Master : David Pichardie, Preparation of Agregation exam, 70h, M2, ENS Rennes, France

Master : Thomas Jensen, Software Security, 21h, M2, Université Rennes 1, France.

7.2.2. Supervision

PhD : Yon Fernandez De Retana, Toward verified compilation of Sea of Nodes : semantic properties and reasoning, Université Rennes 1, defended 5 July 2018, Delphine Demange and David Pichardie

PhD : Alix Trieu, Verifying constant-time implementations in a verified compilation toolchain, Université Rennes 1, defended 4 December 2018, Sandrine Blazy and David Pichardie

PhD: Gurvan Cabon, Analyse non locale certifiée en JavaScript grâce à une sémantique annotée, Université Rennes 1, defended 14 December 2018, Alan Schmitt

PhD in progress : Timothée Haudebourg, Verification of Higher-Order Functional Programs using Tree Automata, September 2017, Thomas Genet and Thomas Jensen

PhD in progress : Rémi Hutin, A C compiler ensuring security properties, September 2018, Sandrine Blazy and David Pichardie

7.2.3. Juries

- Sandrine Blazy, jury member for the selection of CNRS CR and DR (researchers) candidates, February and March 2018, CNRS, Paris, France.
- Sandrine Blazy, jury member for the promotion to a senior lecturer position at the University of Kent, May 2018, Canterbury, Great Britain
- Sandrine Blazy, jury member (reviewer) for the PhD defense of Martin Clochard, March 2018, Paris-Sud University
- Sandrine Blazy, jury member for the PhD defense of Huisong Li, March 2018, ENS, Paris
- Sandrine Blazy, jury member for the HDR defense of Aurélie Hurault, July 2018, Université de Toulouse
- Sandrine Blazy, jury member of the GDR GPL PhD award committee.
- Thomas Jensen, president of jury for the PhD defense of Yon Fernandez de Retana, July 2018, University of Rennes 1.
- Alan Schmitt, jury member (reviewer) for the PhD defense of Guillaume Claret, September 2018, Université Sorbonne Paris Cité
- Sandrine Blazy, jury member for the PhD defense of Rabab Bouziane, December 2018, Université Rennes 1
- David Pichardie, jury member (reviewer) for the PhD defense of Jiangchao Liu, February 2018, Paris Sciences et Lettres Research University
- David Pichardie, jury member (reviewer) for the PhD defense of Thibaut Girka, July 2018, Université Paris-Diderot
- David Pichardie, jury member (reviewer) for the PhD defense of Jean-Christophe Léchenet, July 2018, Université Paris-Saclay
- David Pichardie, jury member (reviewer) for the PhD defense of Sigurd Schneider, November 2018, Saarland University, Germany
- David Pichardie, jury member (reviewer) for the PhD defense of Narjes Jomaa, December 2018, Université de Lille
- Frédéric Besson, member of hiring committee for a Maître de Conférence position, Spring 2018, ENSIMAG

7.3. Popularization

7.3.1. Interventions

- Thomas Genet gave a talk “Bug, Virus, Intrusion, Pirates... So many threats and no defense? Yes... maths.” in high schools close to Rennes. 2018.
- David Cachera contributed to the animation of “École Médiation scientifique en informatique ”, Société Informatique de France, June 2018.

8. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] G. Cabon. *Non Local Analyses Certification With an Annotated Semantics*, Université Rennes 1, December 2018, https://hal.inria.fr/tel-01978292

[2] Y. Fernández de Retana. *Toward verified compilation of Sea of Nodes : semantic properties and reasoning*, Université Rennes 1, July 2018, https://tel.archives-ouvertes.fr/tel-01865395

[3] A. Trieu. *Verifying Constant-Time Implementations in a Verified Compilation Toolchain*, Université Rennes 1, December 2018, https://hal.inria.fr/tel-01944510

Articles in International Peer-Reviewed Journals

[4] M. Bodin, P. Gardner, T. Jensen, A. Schmitt. *Skeletal Semantics and their Interpretations*, in "Proceedings of the ACM on Programming Languages", 2019, vol. 44, pp. 1-31 [*DOI :* 10.1145/3290357], https://hal.inria.fr/hal-01881863

[5] A. Bonifati, S. Dumbrava. *Graph Queries: From Theory to Practice*, in "ACM SIGMOD Record", December 2018, vol. 47, n^{o} 4, https://hal.inria.fr/hal-01977048

[6] Y. Zakowski, D. Cachera, D. Demange, G. Petri, D. Pichardie, S. Jagannathan, J. Vitek. *Verifying a Concurrent Garbage Collector with a Rely-Guarantee Methodology*, in "Journal of Automated Reasoning", November 2018 [*DOI :* 10.1007/s10817-018-9489-x], https://hal.archives-ouvertes.fr/hal-01897251

International Conferences with Proceedings

[7] F. Besson, A. Dang, T. Jensen. *Securing Compilation Against Memory Probing*, in "PLAS '18 - 13th Workshop on Programming Languages and Analysis for Security", Toronto, Canada, ACM, October 2018, pp. 29-40 [*DOI :* 10.1145/3264820.3264822], https://hal.inria.fr/hal-01901765

[8] F. Besson, T. Jensen, J. Lepiller. *Modular Software Fault Isolation as Abstract Interpretation*, in "SAS 2018 - 25th International Static Analysis Symposium", Freiburg, Germany, LNCS, Springer, August 2018, vol. 11002, pp. 166-186 [*DOI :* 10.1007/978-3-319-99725-4_12], https://hal.inria.fr/hal-01894116

[9] S. Blazy, R. Hutin. *Formal Verification of a Program Obfuscation Based on Mixed Boolean-Arithmetic Expressions*, in "CPP - Certified Proofs and Programs", Cascais, Portugal, January 2019 [*DOI :* 10.1145/3293880.3294103], https://hal.inria.fr/hal-01955773

[10] A. Charguéraud, A. Schmitt, T. Wood. *JSExplain: A Double Debugger for JavaScript*, in "The Web Conference 2018", Lyon, France, April 2018, pp. 1-9 [*DOI :* 10.1145/3184558.3185969], https://hal.inria.fr/hal-01745792

[11] D. Demange, Y. Fernández de Retana, D. Pichardie. *Semantic reasoning about the sea of nodes*, in "CC 2018 - 27th International Conference on Compiler Construction", Vienna, Austria, ACM Press, February 2018, pp. 163-173 [*DOI :* 10.1145/3178372.3179503], https://hal.inria.fr/hal-01723236

[12] T. Genet. *Automata and Equations based Approximations for Reachability Analysis*, in "WRLA 2018 - 12th International Workshop on Rewriting Logic and its Applications", Thessalonique, Greece, April 2018, 1 p. , Invited talk, https://hal.archives-ouvertes.fr/hal-01775202

[13] T. Genet. *Completeness of Tree Automata Completion*, in "FSCD 2018 - 3rd International Conference on Formal Structures for Computation and Deduction", Oxford, United Kingdom, July 2018, pp. 1-20 [*DOI :* 10.4230/LIPIcs.FSCD.2018.15], https://hal.archives-ouvertes.fr/hal-01778407

[14] T. GENET, T. GILLARD, T. HAUDEBOURG, S. L. CONG. *Extending Timbuk to Verify Functional Programs*, in "WRLA 2018 - 12th International Worshop on Rewriting Logic and its Applications", Thessalonique, Greece, IEEE, April 2018, pp. 153-163 [*DOI :* 10.1007/978-3-319-99840-4_9], https://hal.archives-ouvertes.fr/hal-01775190

[15] T. GENET, T. HAUDEBOURG, T. JENSEN. *Verifying Higher-Order Functions with Tree Automata*, in "FoSSaCS 2018 - 21st International Conference on Foundations of Software Science and Computation Structures", Thessalonique, Greece, Springer, April 2018, pp. 565-582 [*DOI :* 10.1007/978-3-319-89366-2_31], https://hal.archives-ouvertes.fr/hal-01775188

[16] T. GENET, T. HAUDEBOURG, T. JENSEN. *Vérifier des fonctions d'ordre supérieur à l'aide d'automates d'arbre*, in "17èmes Journées AFADL 2018 - Approches Formelles dans l'Assistance au Développement de Logiciels", Grenoble, France, May 2018, pp. 1-3, https://hal.inria.fr/hal-01790916

[17] D. KÄSTNER, J. BARRHO, U. WÜNSCHE, M. SCHLICKLING, B. SCHOMMER, M. SCHMIDT, C. FERDINAND, X. LEROY, S. BLAZY. *CompCert: Practical Experience on Integrating and Qualifying a Formally Verified Optimizing Compiler*, in "ERTS2 2018 - 9th European Congress Embedded Real-Time Software and Systems", Toulouse, France, 3AF, SEE, SIE, January 2018, pp. 1-9, https://hal.inria.fr/hal-01643290

[18] S. LENGLET, A. SCHMITT. *HOπ in Coq*, in "CPP 2018 - The 7th ACM SIGPLAN International Conference on Certified Programs and Proofs", Los Angeles, United States, January 2018, 14 p. [*DOI :* 10.1145/3167083], https://hal.inria.fr/hal-01614987

[19] *Best Paper*
A. SALIM AL-SIBAHI, A. S. DIMOVSKI, T. JENSEN, A. WASOWSKI. *Verification of High-Level Transformations with Inductive Refinement Types*, in "GPCE 2018 - 17th International Conference on Generative Programming: Concepts & Experience", Boston, United States, November 2018, pp. 1-14, https://hal.inria.fr/hal-01898058.

[20] Y. ZAKOWSKI, D. CACHERA, D. DEMANGE, D. PICHARDIE. *Verified Compilation of Linearizable Data Structures: Mechanizing Rely Guarantee for Semantic Refinement*, in "SAC 2018 - The 33rd ACM/SIGAPP Symposium On Applied Computing", Pau, France, ACM, April 2018, pp. 1881-1890 [*DOI :* 10.1145/3167132.3167333], https://hal.archives-ouvertes.fr/hal-01653620

References in notes

[21] C. CLICK, M. PALECZNY. *A Simple Graph-based Intermediate Representation*, in "Papers from the 1995 ACM SIGPLAN Workshop on Intermediate Representations", New York, NY, USA, IR '95, ACM, 1995, pp. 35–49, http://doi.acm.org/10.1145/202529.202534

[22] Y. ZAKOWSKI, D. CACHERA, D. DEMANGE, G. PETRI, D. PICHARDIE, S. JAGANNATHAN, J. VITEK. *Verifying a Concurrent Garbage Collector using a Rely-Guarantee Methodology*, in "ITP 2017 - 8th International Conference on Interactive Theorem Proving", Brasília, Brazil, Lecture Notes in Computer Science, Springer, September 2017, vol. 10499, pp. 496-513 [*DOI :* 10.1007/978-3-319-66107-0_31], https://hal.inria.fr/hal-01613389