RESEARCH CENTRE

Rennes - Bretagne Atlantique

IN PARTNERSHIP WITH:

Université Rennes 1, École normale supérieure de Rennes

2021

ACTIVITY REPORT

Project-Team

CELTIQUE

Software certification with semantic analysis

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

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Contents

Project-Team CELTIQUE 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 3
- 2.1 Project overview . . . 3

3 Research program 3

4 Application domains 3

5 New software and platforms 3
- 5.1 New software . . . 4
  - 5.1.1 CompcertSSA . . . 4
  - 5.1.2 Timbuk . . . 4
  - 5.1.3 jsexplain . . . 4
  - 5.1.4 JSCert . . . 5
  - 5.1.5 necro . . . 5
  - 5.1.6 Causality . . . 5
  - 5.1.7 itauto . . . 5

6 New results 6
- 6.1 Skeletal Semantics . . . 6
- 6.2 Control-Flow Analyses for Higher-Order Languages . . . 6
- 6.3 Constant-time verification by compilation . . . 7
- 6.4 Formally Verified Speculation and Deoptimization in a JIT Compiler . . . 7
- 6.5 Geometry of Causality . . . 8
- 6.6 Itauto: a reflexive tactic for intuitionistic propositional logic . . . 8
- 6.7 Verification of Program Transformations with Inductive Refinement Types . . . 8
- 6.8 Verified Functional Programming of an Abstract Interpreter . . . 9

7 Partnerships and cooperations 9
- 7.1 International initiatives . . . 9
  - 7.1.1 Participation in other International Programs . . . 9
- 7.2 European initiatives . . . 9
  - 7.2.1 FP7 & H2020 projects . . . 9
- 7.3 National initiatives . . . 10
  - 7.3.1 The ANR Scrypt project . . . 10
  - 7.3.2 The ANR CISC project . . . 10
- 7.4 Regional initiatives . . . 10
  - 7.4.1 Labex Combinlabs SCRATCHS project . . . 10

8 Dissemination 11
- 8.1 Promoting scientific activities . . . 11
  - 8.1.1 Scientific events: organisation . . . 11
  - 8.1.2 Scientific events: selection . . . 11
  - 8.1.3 Journal . . . 11
  - 8.1.4 Invited talks . . . 11
  - 8.1.5 Leadership within the scientific community . . . 12
  - 8.1.6 Scientific expertise . . . 12
  - 8.1.7 Research administration . . . 12
- 8.2 Teaching - Supervision - Juries . . . 12
  - 8.2.1 Teaching . . . 12
  - 8.2.2 Supervision . . . 13
  - 8.2.3 Juries . . . 14

9 Scientific production **15**
9.1 Major publications 15
9.2 Publications of the year 15

Project-Team CELTIQUE

Creation of the Project-Team: 2009 July 01

Keywords

Computer sciences and digital sciences

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

A2.4.1. – Analysis

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 members, visitors, external collaborators

Research Scientists

- Thomas Jensen [Team leader, Inria, Senior Researcher, HDR]
- Frédéric Besson [Inria, Researcher]
- Simon Castellan [Inria, Researcher]
- Benoît Montagu [Inria, Starting Research Position]
- Alan Schmitt [Inria, Senior Researcher, HDR]

Faculty Members

- Sandrine Blazy [Univ de Rennes I, Professor, HDR]
- David Cachera [École normale supérieure de Rennes, Associate Professor, HDR]
- Delphine Demange [Univ de Rennes I, Associate Professor]
- Thomas Genet [Univ de Rennes I, Associate Professor, HDR]
- Rémi Hutin [École normale supérieure de Rennes, From Sep 2021]
- David Pichardie [École normale supérieure de Rennes, Professor, Until September 2021, HDR]

PhD Students

- Guillaume Ambal [Univ de Rennes I]
- Guillaume Barbier [Univ de Rennes I, from Oct 2021]
- Aurele Barriere [École normale supérieure de Rennes]
- Santiago Bautista [École normale supérieure de Rennes]
- Jean Loup Hatchikian-Houdot [Inria, from Oct 2021]
- Rémi Hutin [École normale supérieure de Rennes, until Aug 2021]
- Adam Khayam [Inria]
- Theo Losekoot [Univ de Rennes I, from Sep 2021]
- Solene Mirliaz [École normale supérieure de Rennes]
- Louis Noizet [Univ de Rennes I]
- Gautier Raimondi [Inria]
- Vincent Rebiscoul [Univ de Rennes I]

Interns and Apprentices

- Guillaume Barbier [École normale supérieure de Rennes, from Feb 2021 until Jul 2021]
- Arnaud Daby Seesaram [Ecole normale supérieure Paris-Saclay, from Jun 2021 until Jul 2021]
- Baptiste Demoussel [Inria, from May 2021 until Jul 2021]
- Theo Gouzien [Univ de Rennes I, from Feb 2021 until Jul 2021]
- Olivier Idir [Inria, from Apr 2021 until Jul 2021]
- Theo Losekoot [École normale supérieure de Rennes, from Feb 2021 until Jul 2021]

Administrative Assistant

- Stephanie Gosselin Lemaile [Inria]

Visiting Scientist

- Yann Herklotz [Imperial College London, from May 2021 until Aug 2021]

External Collaborators

- Mickael Delahaye [DGA]
- Emmanuel Fleury [Univ de Bordeaux]

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.

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

3 Research program

The Celtique team conducts research in

- mechanised semantics of programming languages,
- semantics-based program analysis,
- certified compilation,
- language-based software security.

4 Application domains

We work with three application domains: Java software for small devices, embedded C programs, and web applications.

5 New software and platforms

5.1 New software

5.1.1 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.

URL: `https://compcertssa.gitlabpages.inria.fr/`

Publications: hal-01378393, hal-01193281, hal-01110783, hal-01097677, hal-01110779

Contact: Delphine Demange

Participants: Sandrine Blazy, Delphine Demange, Yon Fernandez de Retana, David Pichardie, Leo Stefanesco

5.1.2 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.

URL: `http://people.irisa.fr/Thomas.Genet/timbuk/index.html`

Contact: Thomas Genet

Participant: Thomas Genet

5.1.3 jsexplain

Name: 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.

URL: `https://gitlab.inria.fr/star-explain/jsexplain`

Publication: hal-01745792

Contact: Alan Schmitt

Partner: Imperial College London

5.1.4 JSCert

Name: Certified JavaScript

Keywords: JavaScript, Coq, Formalisation

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.

URL: http://jscert.org/

Contact: Alan Schmitt

Participants: Alan Schmitt, Martin Bodin

Partner: Imperial College London

5.1.5 necro

Name: necro

Keywords: Semantics, Programming language, Specification language

Functional Description: The goal of the project is to provide a tool to manipulate skeletal semantics, a format to represent the semantics of programming languages.

URL: http://skeletons.inria.fr/necro.html

Contact: Alan Schmitt

5.1.6 Causality

Keywords: Semantics, Interpreter, Concurrency, Programming language

Functional Description: Causality is library to write causal semantics of programming languages, based on a monad for truly concurrent computations. It comes with an implementation of an interpreter for a concurrent variant of OCaml.

Contact: Simon Castellan

5.1.7 itauto

Keyword: Automated theorem proving

Functional Description: itauto is a Coq reflexive tactic for solving intuitionistic propositional logic. The tactic inherits features found in modern SAT solvers: definitional conjunctive normal form, lazy unit propagation and conflict driven backjumping. It is also parametrised by a user-provided tactic that is called on the leaves of the proof search.

URL: https://gitlab.inria.fr/fbesson/itauto

Publication: hal-03508736

Contact: Frederic Besson

Participant: Frederic Besson

6 New results

6.1 Skeletal Semantics

Participants: Guillaume Ambal, Olivier Idir, Thomas Jensen, Adam Khayam, Louis Noizet, Vincent Rébiscoul, Alan Schmitt.

The work on skeletal semantics [6], a modular and formal way to describe semantics or programming languages, has continued to intensify during 2021. Links to papers and tools can be found at skeletons.inria.fr. Several interns, PhD students, and postdocs are also working on skeletal semantics.

Louis Noizet is the main designer of Skel, the skeletal semantics language, and the main developer of `necro`, a tool to manipulate skeletal semantics. Skel has changed less this year as it is maturing. The `necro` tool is now split in a library, an OCaml interpreter generator, and a generator of deep embeddings in Coq.

Guillaume Ambal is studying the inter-derivation of multiple semantics from a given language written in Skel. He has formalized in Coq a meta-semantics for skeletal programs that corresponds to an abstract machine. This semantics is proven correct in relation to the usual big-step semantics, and an OCaml interpreter can be extracted from it. This thus provides a way of creating a certified interpreter for any arbitrary language expressed in Skel. This work has been accepted for publication at CPP 2022 [10].

Adam Khayam is writing a formal semantics of JavaScript to validate that our approach scales to complex and sizable semantics while remaining close to the specification and usable for formal proofs. The initial part of this work has been published in 2021 [15]. We have since identified a major problem with the specification of JavaScript: it implicitly uses delimited computations. We have experimented with a continuation monad in Skel to capture this behavior. A paper describing our approach is being written. In addition, Adam has written the semantics of Webi, a language describing web services and IOT devices, in Skel.

Vincent Rébiscoul is working on static analyses for skeletal semantics. He is designing a framework that can automatically derive a control-flow analysis from the definition of a language as a skeletal semantics. The goal of the approach is to automatically derive the correctness of the analysis from the correctness of its components.

Olivier Idir did a four months internship, supervised by Louis Noizet and Alan Schmitt. He showed how skeletal semantics can be used in the setting of a course in advanced semantics, showing for instance the equivalence of big-step and small-step semantics for a toy language.

6.2 Control-Flow Analyses for Higher-Order Languages

Participants: Benoît Montagu, Thomas Jensen.

The purpose of Control-Flow Analyses (CFA) is to determine which functions can be called at a given call site of a program, and to which arguments a function may have been applied. The results of a CFA provide useful information to optimizing compilers, and to static analyzers for higher-order languages and object oriented languages. CFAs received a lot of attention by the research community, since their introduction in the 90s. In particular, researchers have studied the now standard k-CFA analysis, that over-approximates the behavior of programs by being sensitive to the last k call sites.

Benoît Montagu and Thomas Jensen have shown, using standard abstract interpretation techniques, that k-CFA can be obtained via a series of abstractions from a collecting semantics, that collects traces of control-flow events. The recorded events describe which β-reductions have been performed during the reduction of a λ-term, and which values were produced at every program point. The events and the reduction semantics are carefully designed, so that the events contain the contexts in which they have been created. This contextual information is essential, as the *last k call sites* sensitivity of k-CFA is an abstraction of this contextual information.

Additional results in this work include:

- A novel sensitivity policy, called $k^\star$, that keeps at most k occurrences of call sites, and that is experimentally more precise and has a cost that is similar to the standard *last k call sites* policy.
- A novel abstract domain that represents sets of values of the λ-calculus, and that is equipped with a widening operator.
- A novel static analysis for higher-order languages, called ∇-CFA, that is a CFA based on widening to ensure the convergence of the analyzer. ∇-CFA is able to exploit arithmetic domains whose heights are infinite (e.g., the domain of intervals), whereas standard CFAs are restricted to finite-height domains to ensure convergence.
- A prototype implementation of ∇-CFA and a suite of benchmark programs are publically available both as source code and as a web page that embeds the analyzer.

This work has been published in [14].

6.3 Constant-time verification by compilation

Participant: Sandrine Blazy, David Pichardie, Rémi Hutin.

Timing side-channels are arguably one of the main sources of vulnerabilities in cryptographic implementations. One effective mitigation against timing side-channels is to write programs that do not perform secret-dependent branches and memory accesses. This mitigation, known as "cryptographic constant-time", is adopted by several popular cryptographic libraries.

Observational non-interference (ONI) is a generic information-flow policy for side-channel leakage. Informally, a program is ONI-secure if observing the program leakage during execution does not reveal any information about secrets. Formally, ONI is parametrized by a leakage function l, and different instances of ONI can be recovered through different instantiations of l. One popular instance of ONI is the cryptographic constant-time (CCT) policy, which is widely used in cryptographic libraries to protect against timing and cache attacks. Informally, a program is CCT-secure if it does not branch on secrets and does not perform secret-dependent memory accesses. Another instance of ONI is the constant-resource (CR) policy, a relaxation of the CCT policy which is used in Amazon's s2n implementation of TLS and in several other security applications. Informally, a program is CR-secure if its cost (modelled by a tick operator over an arbitrary semi-group) does not depend on secrets.

In this work, we consider the problem of preserving ONI by compilation. Prior work on the preservation of the CCT policy develops proof techniques for showing that main compiler optimisations preserve the CCT policy. However, these proof techniques critically rely on the fact that the semi-group used to model leakage satisfies the property:

$$l_1 + l_1' = l_2 + l_2' \Longrightarrow l_1 = l_2 \wedge l_1' =_2'$$

Unfortunately, this non-cancelling property fails for the CR policy, because its underlying semi-group is $(\mathbb{N}, +)$ and it is currently not known how to extend existing techniques to policies that do not satisfy non-cancellation.

We propose a methodology for proving the preservation of the CR policy during a program transformation. We present an implementation of some elementary compiler passes, and apply the methodology to prove the preservation of these passes. Our results have been mechanically verified using the Coq proof assistant.

This work has been published in [11].

6.4 Formally Verified Speculation and Deoptimization in a JIT Compiler

Participant: Sandrine Blazy, David Pichardie, Aurèle Barrière.

Just-in-time compilers for dynamic languages routinely generate code under assumptions that may be invalidated at run-time. This allows for specialization of program code to the common case in order to avoid unnecessary overheads due to uncommon cases. This form of software speculation requires support for deoptimization when some of the assumptions fail to hold. In this work we define a model just-in-time compiler with an intermediate representation that makes explicit the synchronization points used for deoptimization and the assumptions made by the compiler's speculation. We also define several common compiler optimizations that can leverage speculation to generate improved code. The optimizations are proved correct with the help of a proof assistant. While our work stops short of proving native code generation, we demonstrate how the verified optimization can be used to obtain significant speed ups in an end-to-end setting.

This work has been published in [8].

6.5 Geometry of Causality

Participant: Simon Castellan.

We provide an interpretation of concurrent open programs in terms of (coloured) Petri Nets. Petri Nets are causal models of concurrency that generalise finite state automata, and are widely used in the verification of distributed systems.

Our interpretation is based on the geometry of interaction methodology where sequential programs are interpreted by sequential token machines. We show that our interpretation is correct in two senses: first its notion of may convergence coincides with the traditional one derived from the standard operational semantics; and second we define an unfolding of our coloured Petri nets in terms of event structures and show that the unfolding of the net obtained from a term coincides with the interpretation of that term in event structures using concurrent games.

This is joint work with Pierre Clairambault (CNRS).

The work opens up an avenue of research into program verification using causality. In this direction, we have implemented our model into an interactive web application.

6.6 Itauto: a reflexive tactic for intuitionistic propositional logic

Participant: Frédéric Besson.

ITAUTO is a Coq reflexive tactic for intuitionistic propositional logic [12]. The tactic inherits features found in modern SAT solvers: definitional conjunctive normal form, lazy unit propagation and conflict driven backjumping. Formulae are hash-consed using native integers thus enabling a fast equality test and a pervasive use of Patricia Trees. ITAUTO proposes a hybrid proof by reflection scheme. The extracted solver calls user-defined tactics on the leaves of the propositional proof search thus enabling theory reasoning and the generation of conflict clauses. The solver has decent efficiency and is more scalable than existing tactics.

6.7 Verification of Program Transformations with Inductive Refinement Types

Participant: Thomas Jensen.

High-level transformation languages like Rascal include expressive features for manipulating large abstract syntax trees: first-class traversals, expressive pattern matching, backtracking, and generalized iterators. We present the design and implementation of an abstract interpretation tool 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. Finally, we evaluate the tool on a series of transformations (normalization, desugaring, refactoring, code generators, type inference, etc.) showing that we can effectively verify stated properties. This work was published in ACM ToSEM [9].

6.8 Verified Functional Programming of an Abstract Interpreter

Participant: David Pichardie.

Interpreters are complex pieces of software: even if the abstract interpretation theory and companion algorithms are well understood, their implementations are subject to bugs, that might question the soundness of their computations. While some formally verified abstract interpreters have been written in the past, writing and understanding them requires expertise in the use of proof assistants, and requires a non-trivial amount of interactive proofs. This paper presents a formally verified abstract interpreter fully programmed and proved correct in the F* verified programming environment. Thanks to F* refinement types and SMT prover capabilities we demonstrate a substantial saving in proof effort compared to previous works based on interactive proof assistants. Almost all the code of our implementation, proofs included, written in a functional style, are presented directly in the paper [13].

7 Partnerships and cooperations

7.1 International initiatives

7.1.1 Participation in other International Programs

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. We work with Małgorzata Biernacka and Dariusz Biernacki on formal transformations of operational semantics.

7.2 European initiatives

7.2.1 FP7 & H2020 projects

The SPARTA cybersecurity competence network

Participants: Thomas Jensen, Frédéric Besson.

SPARTA is a Cybersecurity Competence Network, supported by the EU's H2020 program, with the objective to develop and implement top-tier research and innovation collaborative actions. Guided by concrete challenges forming an ambitious Cybersecurity Research & Innovation Roadmap, SPARTA aims to set up unique collaboration means, leading the way in building transformative capabilities and forming a world-leading Cybersecurity Competence Network across the EU. The SPARTA consortium assembles 44 actors from 14 EU Member States at the intersection of scientific excellence, technological innovation, and societal sciences in cybersecurity.

Celtique is coordinating the Inria participation in the SPARTA network. The team contributes to the program on intelligent infrastructures with techniques for building security-enhanced systems code that respects strong information flow constraints. The team is also leading the elaboration of the SPARTA scientific roadmap, in collaboration with TU Munich.

The ERC VESTA project

Participant: Sandrine Blazy, Aurèle Barrière, Remi Hutin, Solène Mirliaz, David Pichardie.

Keywords: Security, Secure compilation.

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 certifies many advanced abstract interpretation techniques. 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 years. The project started in September 2018 and ended in August 2021 with the departure of David Pichardie.

7.3 National initiatives

7.3.1 The ANR Scrypt project

Participants: Frédéric Besson, Sandrine Blazy, Thomas Jensen, David Pichardie, Remi Hutin.

Keywords: Security, Secure compilation.

The Scrypt project (ANR-18-CE25-0014) aims at providing secure implementations of cryptographic primitives using formal methods and secure compilation techniques. One specific goal is to design secure compilers which preserve the security of the source code against side-channel attacks.

This is a joint project with the Inria team MARELLE, École Polytechnique and the company AMOSSYS.

7.3.2 The ANR CISC project

Participant: 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.

7.4 Regional initiatives

7.4.1 Labex Combinlabs SCRATCHS project

Participant: Frédéric Besson, Jean-Loup Hatchikian-Houdot.

The goal of SCRATCHS (2021-2024) is to co-design a compiler toolchain and a RISC-V micro-controller in order to ensure the absence of side-channel timing leaks. The CELTIQUE team will work at exploiting the security mechanisms in a dedicated secure compiler toolchain.

SCRATCHS is a joint project between the CELTTIQUE team, the CIDRE team and the Lab-Sticc ARCAD team

8 Dissemination

Participants: Frédéric Besson, Sandrine Blazy, Simon Castellan, Delphine Demange, Thomas Genet, Thomas Jensen, Benoît Montagu, David Pichardie, Alan Schmitt.

8.1 Promoting scientific activities

8.1.1 Scientific events: organisation

Member of the organizing committees

- Sandrine Blazy and Thomas Jensen: steering committee of SAS
- Delphine Demange: Steering Committee of CC (2021-2024)

8.1.2 Scientific events: selection

Chair of conference program committees

- Delphine Demange: CC ’21 (Program co-Chair with Rajiv Gupta, UC Riverside). Proceedings: see [17].

Member of the conference program committees

- Sandrine Blazy: AFADL’20, FMTea’21, Types’21, ESOP’21, POPL’22
- Delphine Demange: PriSC ’21
- Benoît Montagu: ML Workshop ’21

Reviewer

- Benoît Montagu: TYPES ’20, ICALP ’21

8.1.3 Journal

Reviewer - reviewing activities

- Alan Schmitt : JFP, JLAMP, TCS, TOCL, TOPLAS
- Frédéric Besson: CESAR, TCAD

8.1.4 Invited talks

- Sandrine Blazy, "A tutorial on teaching deductive verification with Why3", Tutorial Series of the FME Teaching Committee, 09/21
- Delphine Demange, member of faculty panel on "Doing Research in PL", PLMW@POPL ’21

8.1.5 Leadership within the scientific community

- Sandrine Blazy was member (until August 2021) of Section 6 of the national committee for scientific research CoNRS.
- Sandrine Blazy is member of IFIP WG 2.11 on program generation and of IFIP WG 1.9/2.15 on verified software.

8.1.6 Scientific expertise

- Alan Schmitt : Expertise Allocation Spécifique Normalien 2021
- Sandrine Blazy is member of the international scientific advisory board of the Flemish cybersecurity program.
- Sandrine Blazy: expertise for FNR (Luxemburg) and for NSERC (Canada).

8.1.7 Research administration

- Sandrine Blazy is deputy director of IRISA (CNRS UMR 6074).
- Sandrine Blazy is member of the GDR GPL scientific committee

8.2 Teaching - Supervision - Juries

8.2.1 Teaching

- License : Frédéric Besson, Functional programming, 28h, Insa3, Insa Rennes, France
- Licence : Sandrine Blazy, Programmation de confiance, 81h, L3, Université Rennes 1, France
- Master : Sandrine Blazy, Semantics of Programming Languages, 20h, M1, Université Rennes 1, France
- Licence : Delphine Demange, Spécialité Informatique 1 - Algorithmique et Complexité Expérimentale, 40h, L1, Université Rennes 1, France
- Licence : Delphine Demange, Informatique 1 - Programmation Impérative en Java, 40h, L1, Université Rennes 1, France
- Licence : Delphine Demange, Complément en Informatique 1 - Programmation Impérative en Java, 12h, L1, Université Rennes 1, France
- Licence : Delphine Demange, Mise à Niveau en Informatique - Programmation en Java, 12h, 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 : Alan Schmitt, Programmation de Confiance, 30h, L3, Université Rennes 1, France
- Licence : Benoît Montagu, Programmation de Confiance, 20h, L3, Université Rennes 1, France
- Master : Thomas Genet, Analyse et conception formelle, 65h, M1, Université Rennes 1, France
- Master : Benoît Montagu, Analyse et Conception Formelles, 24h, M1, Université Rennes 1, France
- Master : Alan Schmitt, Advanced Semantics, 30h, M2, ENS Rennes, France
- Master : Alan Schmitt, Preparation of Agregation exam, 22h, M2, ENS Rennes, France

- License : Thomas Jensen, Logic, 20h, Insa3, Insa Rennes, France
- Master : Thomas Jensen, Software Security , 14h, M2, Université Rennes 1, France
- Master : Thomas Jensen, Software security, 14h, University of Copenhagen, Denmark
- Master : Simon Castellan, Software Security , 6h, M2, Université Rennes 1, France
- Master: Benoît Montagu, Paradigmes de programmation (préparation à l'agrégation d'informatique), 12h, ENS Rennes, France

8.2.2 Supervision

- PhD in progress: Adam Khayam, Formal Semantics of Multitier Languages, July 2019, Alan Schmitt
- PhD in progress: Guillaume Ambal, Sémantiques Squelettiques pour Calculs de Processus, September 2019, Alan Schmitt
- PhD in progress: Théo Losekoot, Vérification automatique de propriétés structurelles et relationnelles, September 2021, Thomas Genet and Thomas Jensen
- PhD in progress: Louis Noizet, Compilation Certifiée de Sémantiques Squelettiques, October 2019, Alan Schmitt
- PhD in progress: Vincent Rébiscoul, Analyses Statiques pour Sémantiques Squelettiques, September 2020, Thomas Jensen and Alan Schmitt
- PhD in progress: Gautier Raimondi, Secure Compilation against Side-channel Attacks, September 2020, Thomas Jensen and Frédéric Besson.
- PhD in progress: Jean-Loup Hatchikian-Houdot, Security-enhancing compiler against side-channel attacks, Octobre 2021, Guillaume Hiet and Frédéric Besson
- PhD in progress: Shenghao Yuan, Verified programming and secure integration of operating system libraries, Octobre 2020, Jean-Pierre Talpin and Frédéric Besson
- PhD in progress: Santiago Bautista, Static analyses for semi-interactive program verification, September 2020, Thomas Jensen and Benoît Montagu
- PhD: Rémi Hutin, A C compiler ensuring security properties, defended December 2021, Sandrine Blazy and David Pichardie
- PhD in progress : Aurèle Barrière, Formal verification of a JIT compiler, September 2019, Sandrine Blazy and David Pichardie
- PhD in progress : Solène Mirliaz, Provable Cost Analysis of Pipelined-Optimized Cryptographic Code, September 2019, Sandrine Blazy and David Pichardie
- Visiting PhD student: Yann Herklotz, Formal Verification of the Gated SSA form in CompCert, May 2021 to July 2021, Sandrine Blazy and Delphine Demange
- Master 2 internship: Théo Gouzien, Formalizing the Monadic Gated SSA form in Coq, February 2021 to July 2021, Delphine Demange
- Master 1 internship: Olivier Idir, Advanced Semantics Course in Skel, April 2021 to July 2021
- Licence 3 internship: Arnaud Daby-Seesaram, Iterations Strategies for Dataflow Analyses in CompCert, May 2021 to July 2021, Sandrine Blazy and Delphine Demange

8.2.3 Juries

PhD and HDR defenses:

- Alan Schmitt, jury member (reviewer) for the PhD defense of Martin Vassor, May 2021, Université Grenoble Alpes
- Alan Schmitt, jury member for the PhD defense of Raphaël Monat, November 2021, Sorbonne Université
- Delphine Demange, jury member for the PhD defense of Cyril Six, July 2021, Université Grenoble Alpes
- Sandrine Blazy, jury member (reviewer) for the PhD defense of Vincent Iampietro, U. Montpellier. 12/2021
- Sandrine Blazy, jury member (president) for the PhD defense of Lucas Franceschino, U. Rennes, 12/21.
- Sandrine Blazy, jury member for the PhD defense of Son Tuan Vu, Sorbonne university, 04/21.
- Sandrine Blazy, jury member (reviewer) for the HDR defense of Pierre-Évariste Dagand, Sorbonne U., 03/21.
- Sandrine Blazy, jury member (reviewer) for the HDR defense of Sylvain Boulmé, U. Grenoble Alpes, 07/21.
- Thomas Jensen, president of the jury for the HdR of Guillaume Hiet, Centrale Supelec. 12/21.
- Thomas Jensen, jury member (reviewer) for the PhD defense of Sven Keidel, U. Mainz, Germany.
- Thomas Jensen, jury member (reviewer) for the PhD defense of Martina Olliaro, U. Ca' Foscari, Venice, Italy.
- Thomas Jensen, jury member (reviewer) for the PhD defense of Tom Seed, U. Kent at Canterbury, U.K.
- Thomas Jensen, president of the jury for the PhD defense of Itsaka Rakotonirina, U. of Lorraine.

Hiring and Promotion Committees:

- Sandrine Blazy, jury member for the national selection of CNRS CR and DR (researchers) candidates, Spring 2020, Paris.
- Delphine Demange, member of hiring committee for an Associate Professor, Université de Paris and IRIF, Spring 2021
- Delphine Demange, jury member for Campagne Locale des Moyens Incitatifs, Inria RBA, April 2021
- Thomas Jensen, member of the Promotion Board at Royal Technical University (KTH), Stockholm, Sweden.
- Thomas Jensen was member of the hiring committee in Professor in Cybersecurity, Paris Telecom.
- Thomas Jensen was member of the hiring committe of an Assistant Professor in Software and Systems Security, University of Copenhagen, Denmark.

Others:

- Sandrine Blazy, jury member of the GDR GPL PhD award committee.
- Thomas Jensen, member of the ERCIM Security and Trust WG 2021 PhD award committee.

9 Scientific production

9.1 Major publications

[1] O. Andreescu, T. Jensen, S. Lescuyer and B. Montagu. 'Inferring frame conditions with static correlation analysis'. In: *Proceedings of the ACM on Programming Languages* 3.POPL (Jan. 2019), pp. 1–29. DOI: 10.1145/3290360. URL: https://hal.inria.fr/hal-02413262.

[2] G. Barthe, S. Blazy, B. Grégoire, R. Hutin, V. Laporte, D. Pichardie and A. Trieu. 'Formal verification of a constant-time preserving C compiler'. In: *Proceedings of the ACM on Programming Languages* 4.POPL (Jan. 2020), pp. 1–30. DOI: 10.1145/3371075. URL: https://hal.univ-lorraine.fr/hal-02975012.

[3] G. Barthe, D. Demange and D. Pichardie. 'Formal Verification of an SSA-based Middle-end for CompCert'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 36.1 (2014), p. 35. DOI: 10.1145/2579080. URL: https://hal.inria.fr/hal-01097677.

[4] F. Besson, S. Blazy, A. Dang, T. Jensen and P. Wilke. 'Compiling Sandboxes: Formally Verified Software Fault Isolation'. In: *ESOP 2019 - 28th European Symposium on Programming*. Vol. 11423. LNCS. Prague, Czech Republic: Springer, Apr. 2019, pp. 499–524. DOI: 10.1007/978-3-030-17184-1_18. URL: https://hal.inria.fr/hal-02316189.

[5] M. Bodin, A. Charguéraud, D. Filaretti, P. Gardner, S. Maffeis, D. Naudziuniene, A. Schmitt and G. Smith. 'A Trusted Mechanised JavaScript Specification'. In: *POPL 2014 - 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. San Diego, United States, Jan. 2014. URL: https://hal.inria.fr/hal-00910135.

[6] M. Bodin, P. Gardner, T. Jensen and A. Schmitt. 'Skeletal Semantics and their Interpretations'. In: *Proceedings of the ACM on Programming Languages* 44 (2019), pp. 1–31. DOI: 10.1145/3290357. URL: https://hal.inria.fr/hal-01881863.

[7] J.-H. Jourdan, V. Laporte, S. Blazy, X. Leroy and D. Pichardie. 'A formally-verified C static analyzer'. In: *POPL 2015: 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Mumbai, India: ACM, Jan. 2015, pp. 247–259. DOI: 10.1145/2676726.2676966. URL: https://hal.inria.fr/hal-01078386.

9.2 Publications of the year

International journals

[8] A. Barrière, S. Blazy, O. Flückiger, D. Pichardie and J. Vitek. 'Formally verified speculation and deoptimization in a JIT compiler'. In: *Proceedings of the ACM on Programming Languages* 5.POPL (4th Jan. 2021), p. 26. DOI: 10.1145/3434327. URL: https://hal.archives-ouvertes.fr/hal-03185848.

[9] A. S. Al-Sibahi, T. P. Jensen, A. S. Dimovski and A. Wąsowski. 'Verification of Program Transformations with Inductive Refinement Types'. In: *ACM Transactions on Software Engineering and Methodology* 30.1 (31st Jan. 2021), pp. 1–33. DOI: 10.1145/3409805. URL: https://hal.inria.fr/hal-03518825.

International peer-reviewed conferences

[10] G. Ambal, S. Lenglet and A. Schmitt. 'Certified Abstract Machines for Skeletal Semantics'. In: Certified Programs and Proofs. Philadelphia, United States, 17th Jan. 2022. URL: https://hal.inria.fr/hal-03466807.

[11] G. Barthe, S. Blazy, R. Hutin and D. Pichardie. 'Secure Compilation of Constant-Resource Programs'. In: CSF 2021 - 34th IEEE Computer Security Foundations Symposium. Dubrovnik, Croatia: IEEE, 2021, pp. 1–12. URL: https://hal.archives-ouvertes.fr/hal-03221440.

[12] F. Besson. 'Itauto: An Extensible Intuitionistic SAT Solver'. In: ITP 2021 - 12th International Conference on Interactive Theorem Proving. Roma, Italy, 29th June 2021, pp. 1–18. DOI: 10.4230/LIPIcs.ITP.2021.9. URL: https://hal.inria.fr/hal-03508736.

[13] L. Franceschino, J.-P. Talpin and D. Pichardie. 'Verified Functional Programming of an Abstract Interpreter'. In: SAS 2021 - 28th Static Analysis Symposium. Chicago, United States, 17th Oct. 2021, pp. 1–20. URL: https://hal.inria.fr/hal-03342997.

[14] B. Montagu and T. Jensen. 'Trace-Based Control-Flow Analysis'. In: PLDI 2021 - 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Virtual, Canada: ACM, 20th June 2021, pp. 1–15. DOI: 10.1145/3453483.3454057. URL: https://hal.inria.fr/hal-03266981.

National peer-reviewed Conferences

[15] A. Khayam, L. Noizet and A. Schmitt. 'JSkel: Towards a Formalization of JavaScript's Semantics'. In: JFLA 2021 - Journées Francophones des Langages Applicatifs. Virtuelles, France, 7th Apr. 2021, pp. 1–22. URL: https://hal.inria.fr/hal-03509431.

Scientific book chapters

[16] P. Clairambault, S. Castellan and P. Dybjer. 'Categories with Families: Unityped, Simply Typed, and Dependently Typed'. In: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics.* Vol. 20. Outstanding Contributions to Logic. Springer International Publishing, 21st Mar. 2021, pp. 135–180. DOI: 10.1007/978-3-030-66545-6_5. URL: https://hal.archives-ouvertes.fr/hal-03419296.

Edition (books, proceedings, special issue of a journal)

[17] A. Smith, D. Demange and R. Gupta, eds. *CC 2021: Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction.* CC '21 - 30th ACM SIGPLAN International Conference on Compiler Construction. Virtual Republic of Korea: ACM, Mar. 2021. DOI: 10.1145/3446804. URL: https://hal.inria.fr/hal-03185984.

Reports & preprints

[18] S. Castellan and P. Clairambault. *Disentangling Parallelism and Interference in Game Semantics.* 26th Mar. 2021. URL: https://hal.archives-ouvertes.fr/hal-03182043.

[19] S. Castellan, P. Clairambault and G. Winskel. *The Mays and Musts of Concurrent Strategies.* 23rd Aug. 2021. URL: https://hal.archives-ouvertes.fr/hal-03323995.

[20] P. Clairambault and S. Castellan. *The Concurrent Games Abstract Machine: Multi-Token Geometry of Interaction and its Causal Unfolding.* 15th July 2021. URL: https://hal.archives-ouvertes.fr/hal-03286443.

[21] S. Mirliaz and D. Pichardie. *A Flow-Insensitive-Complete Program Representation.* 19th Oct. 2021. URL: https://hal.archives-ouvertes.fr/hal-03384612.