IN PARTNERSHIP WITH:
Université Rennes 1

École normale supérieure de Rennes

Activity Report 2015

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. - Reliability, certification
2.4.1. - Analysis
2.4.2. - Verification
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 [Normale Sup Rennes, Associate Professor, HdR]
Delphine Demange [Univ. Rennes I, Associate Professor]
Thomas Genet [Univ. Rennes I, Associate Professor, HdR]
Barbara Kordy [INSA Rennes, Associate Professor]
David Pichardie [Normale Sup Rennes, Professor, HdR]
Charlotte Truchet [Univ. Nantes, Associate Professor]

Engineer

Laurent Guillo [CNRS, Senior engineer]

PhD Students

Oana Andreescu [Prove & Run, granted by CIFRE]
Martin Bodin [Univ. Rennes I]
Pauline Bolignano [Prove & Run, granted by CIFRE]

David Buhler [CEA, granted by CEA]
Gurvan Cabon [Inria]
Yon Fernandez de Retana [Univ. Rennes I]
Vincent Laporte [Univ. Rennes I, until Nov 2015, granted by ANR VERASCO project]
Stéphanie Riaud [DGA-MI]
Yann Salmon [High-school teacher]
Florent Saudel [Amossys, from Nov 2015, granted by CIFRE]
Pierre Wilke [Univ. Rennes I]
Yannick Zakowski [Normale Sup Rennes]

Post-Doctoral Fellows

Petar Maksimovic [Inria, until Nov 2015]
Marek Materzok [Inria]

Administrative Assistant

Lydie Mabil [Inria]

Other

Colas Le Guernic [DGA]

2. Overall Objectives

2.1. Project overview

The overall goal of the CELTIQUE project is to improve the security and reliability of software with semantic certification that attests to its well-behavedness. The semantic analyses extract approximate but sound descriptions of software behaviour from which a proof of security can be constructed. The analyses of relevance include numerical data flow analysis, control flow analysis for higher-order languages, alias and points-to analysis for heap structure manipulation, and various kinds of information flow analysis.

To achieve this goal, the project conducts work on improving semantic analysis techniques, as well as work on using proof assistants such as Coq to develop and prove properties of these analyses. 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.

We target 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. Research Program

3.1. Static program analysis

Static program analysis is concerned with obtaining information about the run-time behaviour of a program without actually running it. This information may concern the values of variables, the relations among them, dependencies between program values, the memory structure being built and manipulated, the flow of control, and, for concurrent programs, synchronisation among processes executing in parallel. Fully automated analyses usually render approximate information about the actual program behaviour. The analysis is correct if the information includes all possible behaviour of a program. Precision of an analysis is improved by reducing the amount of information describing spurious behaviour that will never occur.

Static analysis has traditionally found most of its applications in the area of program optimisation where information about the run-time behaviour can be used to transform a program so that it performs a calculation faster and/or makes better use of the available memory resources. The last decade has witnessed an increasing use of static analysis in software verification for proving invariants about programs. The Celtique project is mainly concerned with this latter use. Examples of static analysis include:

- Data-flow analysis as it is used in optimising compilers for imperative languages. The properties can either be approximations of the values of an expression ("the value of variable x is greater than 0" or x is equal to y at this point in the program") or more intensional information about program behaviour such as "this variable is not used before being re-defined" in the classical "dead-variable" analysis [74].
- Analyses of the memory structure includes shape analysis that aims at approximating the data structures created by a program. Alias analysis is another data flow analysis that finds out which variables in a program addresses the same memory location. Alias analysis is a fundamental analysis for all kinds of programs (imperative, object-oriented) that manipulate state, because alias information is necessary for the precise modelling of assignments.
- Control flow analysis will find a safe approximation to the order in which the instructions of a program are executed. This is particularly relevant in languages where parameters or functions can be passed as arguments to other functions, making it impossible to determine the flow of control from the program syntax alone. The same phenomenon occurs in object-oriented languages where it is the class of an object (rather than the static type of the variable containing the object) that determines which method a given method invocation will call. Control flow analysis is an example of an analysis whose information in itself does not lead to dramatic optimisations (although it might enable in-lining of code) but is necessary for subsequent analyses to give precise results.

Static analysis possesses strong **semantic foundations**, notably abstract interpretation [57], that allow to prove its correctness. The implementation of static analyses is usually based on well-understood constraint-solving techniques and iterative fixpoint algorithms. In spite of the nice mathematical theory of program analysis and the solid algorithmic techniques available one problematic issue persists, *viz.*, the *gap* between the analysis that is proved correct on paper and the analyser that actually runs on the machine. While this gap might be small for toy languages, it becomes important when it comes to real-life languages for which the implementation and maintenance of program analysis tools become a software engineering task. A *certified static analysis* is an analysis that has been formally proved correct using a proof assistant.

In previous work we studied the benefit of using abstract interpretation for developing **certified static analyses** [55], [77]. The development of certified static analysers is an ongoing activity that will be part of the Celtique project. We use the Coq proof assistant which allows for extracting the computational content of a constructive proof. A Caml implementation can hence be extracted from a proof of existence, for any program, of a correct approximation of the concrete program semantics. We have isolated a theoretical framework based on abstract interpretation allowing for the formal development of a broad range of static analyses. Several case studies for the analysis of Java byte code have been presented, notably a memory usage analysis [56]. This work has recently found application in the context of Proof Carrying Code and have also been successfully applied to particular form of static analysis based on term rewriting and tree automata [4].

3.1.1. Static analysis of Java

Precise context-sensitive control-flow analysis is a fundamental prerequisite for precisely analysing Java programs. Bacon and Sweeney's Rapid Type Analysis (RTA) [48] is a scalable algorithm for constructing an initial call-graph of the program. Tip and Palsberg [80] have proposed a variety of more precise but scalable call graph construction algorithms *e.g.,* MTA, FTA, XTA which accuracy is between RTA and 0'CFA. All those analyses are not context-sensitive. As early as 1991, Palsberg and Schwartzbach [75], [76] proposed a theoretical parametric framework for typing object-oriented programs in a context-sensitive way. In their setting, context-sensitivity is obtained by explicit code duplication and typing amounts to analysing the expanded code in a context-insensitive manner. The framework accommodates for both call-contexts and allocation-contexts.

To assess the respective merits of different instantiations, scalable implementations are needed. For Cecil and Java programs, Grove *et al.,* [64], [63] have explored the algorithmic design space of contexts for benchmarks of significant size. Later on, Milanova *et. al.,* [71] have evaluated, for Java programs, a notion of context called *object-sensitivity* which abstracts the call-context by the abstraction of the `this` pointer. More recently, Lhotak and Hendren [69] have extended the empiric evaluation of object-sensitivity using a BDD implementation allowing to cope with benchmarks otherwise out-of-scope. Besson and Jensen [53] proposed to use DATALOG in order to specify context-sensitive analyses. Whaley and Lam [81] have implemented a context-sensitive analysis using a BDD-based DATALOG implementation.

Control-flow analyses are a prerequisite for other analyses. For instance, the security analyses of Livshits and Lam [70] and the race analysis of Naik, Aiken [72] and Whaley [73] both heavily rely on the precision of a control-flow analysis.

Control-flow analysis allows to statically prove the absence of certain run-time errors such as "message not understood" or cast exceptions. Yet it does not tackle the problem of "null pointers". Fahnrich and Leino [60] propose a type-system for checking that after object creation fields are non-null. Hubert, Jensen and Pichardie have formalised the type-system and derived a type-inference algorithm computing the most precise typing [67]. The proposed technique has been implemented in a tool called NIT [66]. Null pointer detection is also done by bug-detection tools such as FindBugs [66]. The main difference is that the approach of findbugs is neither sound nor complete but effective in practice.

3.1.2. Quantitative aspects of static analysis

Static analyses yield qualitative results, in the sense that they compute a safe over-approximation of the concrete semantics of a program, w.r.t. an order provided by the abstract domain structure. Quantitative aspects of static analysis are two-sided: on one hand, one may want to express and verify (compute) quantitative properties of programs that are not captured by usual semantics, such as time, memory, or energy consumption; on the other hand, there is a deep interest in quantifying the precision of an analysis, in order to tune the balance between complexity of the analysis and accuracy of its result.

The term of quantitative analysis is often related to probabilistic models for abstract computation devices such as timed automata or process algebras. In the field of programming languages which is more specifically addressed by the Celtique project, several approaches have been proposed for quantifying resource usage: a non-exhaustive list includes memory usage analysis based on specific type systems [65], [47], linear logic approaches to implicit computational complexity [49], cost model for Java byte code [46] based on size relation inference, and WCET computation by abstract interpretation based loop bound interval analysis techniques [58].

We have proposed an original approach for designing static analyses computing program costs: inspired from a probabilistic approach [78], a quantitative operational semantics for expressing the cost of execution of a program has been defined. Semantics is seen as a linear operator over a dioid structure similar to a vector space. The notion of long-run cost is particularly interesting in the context of embedded software, since it provides an approximation of the asymptotic behaviour of a program in terms of computation cost. As for classical static analysis, an abstraction mechanism allows to effectively compute an over-approximation of the semntics, both in terms of costs and of accessible states [54]. An example of cache miss analysis has been developed within this framework [79].

3.1.3. Certified static analysis

In spite of the nice mathematical theory of program analysis (notably abstract interpretation) and the solid algorithmic techniques available one problematic issue persists, *viz.*, the *gap* between the analysis that is proved correct on paper and the analyser that actually runs on the machine. While this gap might be small for toy languages, it becomes important when it comes to real-life languages for which the implementation and maintenance of program analysis tools become a software engineering task.

A *certified static analysis* is an analysis whose implementation has been formally proved correct using a proof assistant. Such analysis can be developed in a proof assistant like Coq [45] by programming the analyser inside the assistant and formally proving its correctness. The Coq extraction mechanism then allows for extracting a Caml implementation of the analyser. The feasibility of this approach has been demonstrated in [6].

We also develop this technique through certified reachability analysis over term rewriting systems. Term rewriting systems are a very general, simple and convenient formal model for a large variety of computing systems. For instance, it is a very simple way to describe deduction systems, functions, parallel processes or state transition systems where rewriting models respectively deduction, evaluation, progression or transitions. Furthermore rewriting can model every combination of them (for instance two parallel processes running functional programs).

Depending on the computing system modelled using rewriting, reachability (and unreachability) permits to achieve some verifications on the system: respectively prove that a deduction is feasible, prove that a function call evaluates to a particular value, show that a process configuration may occur, or that a state is reachable from the initial state. As a consequence, reachability analysis has several applications in equational proofs used in the theorem provers or in the proof assistants as well as in verification where term rewriting systems can be used to model programs.

For proving unreachability, i.e. safety properties, we already have some results based on the over-approximation of the set of reachable terms [61], [62]. We defined a simple and efficient algorithm [59] for computing exactly the set of reachable terms, when it is regular, and construct an over-approximation otherwise. This algorithm consists of a *completion* of a *tree automaton*, taking advantage of the ability of tree automata to finitely represent infinite sets of reachable terms.

To certify the corresponding analysis, we have defined a checker guaranteeing that a tree automaton is a valid fixpoint of the completion algorithm. This consists in showing that for all term recognised by a tree automaton all his rewrites are also recognised by the same tree automaton. This checker has been formally defined in Coq and an efficient Ocaml implementation has been automatically extracted [4]. This checker is now used to certify all analysis results produced by the regular completion tool as well as the optimised version of [50].

4. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Awards

Alan Schmitt has received the 2015 Most Influential POPL Paper Award for the 2005 paper “Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem” [8].

5. New Software and Platforms

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

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

5.3. 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: Frédéric Besson, David Pichardie and Laurent Guillo
- Contact: David Pichardie
- URL: http://sawja.inria.fr/

5.4. SAWJA

Static Analysis Workshop for Java
KEYWORDS: Security - Software - Code review
FUNCTIONAL 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.

- Participants: Frédéric Besson, David Pichardie and Laurent Guillo
- Contact: Frédéric Besson
- URL: http://sawja.inria.fr/

5.5. Timbuk

KEYWORDS: Demonstration - Ocaml - Vérification de programmes - 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/

5.6. 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). 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/

6. New Results

6.1. Certified compilation

We thrive at improving the technology of certified compilation. Our work builds on the infrastructure provided by the CompCert compiler. We are working both at improving the guarantees provided by certified compilation and at formalising state-of-the-art optimisation techniques.

6.1.1. Safer CompCert

Participants: Sandrine Blazy, Frédéric Besson, Pierre Wilke.

The CompCert compiler is proved with respect to an abstract semantics. In previous work [52], we propose a more concrete memory model for the CompCert compiler [68]. This model gives a semantics to more programs and lift the assumption about an infinite memory. This model makes CompCert safer because more programs are captured by the soundness theorem of CompCert and because it allows to reason about memory consumption.

We are investigating the consequences this model on different compiler passes of CompCert [32]. As a sanity check, we prove formally that the existing memory model is an abstraction of our more concrete model thus validating formally the soundness of CompCert's abstract semantics of pointers. We have also port the front-end of the compiler to our new semantics and are working on the compiler back-end.

6.1.2. Verification of optimization techniques

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

The CompCert compiler foregoes using SSA, an intermediate representation employed by many compilers that enables writing simpler, faster optimizers. In previous work [51], we have proposed a formally verified SSA-based middle-end for CompCert, addressing two problems raised by Leroy in 2009: giving an intuitive formal semantics to SSA, and leveraging its global properties to reason locally about program optimizations. Since then, we have studied in more depth the SSA-based optimization techniques with the objective to make the middle-end more realistic, in terms of the efficiency of optimizations, and to rationalize the way the correctness proofs of optimizations are conducted and structured.

First, we have studied in [34] the problem of a verified, yet efficient (i.e. as implemented in production compilers) technique for testing the dominance relation between two nodes in a control flow graph. We propose a formally verified validator of untrusted dominator trees, on top of which we implement and prove correct a fast dominance test.

Second, in [20], we implement and verify two prevailing SSA optimizations (Sparse Conditional Constant Propagation and Global Value Numbering), conducting the proofs in a unique and common sparse optimization proof framework, factoring out many of the dominance-based reasoning steps required in proofs of SSA-based optimizations. Our experimental evaluations indicate both a better precision, and a significant compilation time speedup.

Finally, we have studied (paper under review at the international conference Compiler Construction 2016) the destruction of the SSA form (i.e. at the exit point of the middle-end), a problem that has remained a difficult problem, even in a non-verified environment. We formally defined and proved the properties of the generation of Conventional SSA (CSSA) which make its destruction simple to implement and prove. We implemented and proved correct a coalescing destruction of CSSA, à la Boissinot et al., 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.

6.2. Certified Static Analyses

6.2.1. Certified Analyses for JavaScript

Participants: Martin Bodin, Thomas Jensen, Alan Schmitt.

We have continued our work on the certification of analyses for JavaScript by developing a systematic way to build certified abstract interpreters from big-step semantics using the Coq proof assistant. We based our approach on Schmidt's abstract interpretation principles for natural semantics, and used a pretty-big-step (PBS) semantics, a semantic format proposed by Charguéraud. We proposed a systematic representation of the PBS format and implemented it in Coq. We then showed how the semantic rules can be abstracted in a methodical fashion, independently of the chosen abstract domain, to produce a set of abstract inference rules that specify an abstract interpreter. We proved the correctness of the abstract interpreter in Coq once and for all, under the assumption that abstract operations faithfully respect the concrete ones. We finally showed how to define correct-by-construction analyses: their correction amounts to proving they belong to the abstract semantics. This work has been published at CPP 2015 [19].

In addition, we have worked on hybrid typing of information flow for JavaScript, in collaboration with José Fragoso Santos and Tamara Rezk at Inria Sophia-Antipolis. Our analysis combined static and dynamic typing in order to avoid rejecting programs due to imprecise typing information. This work has been published at TGC 2015 [21].

6.2.2. Certified Analyses for safety-critical C programs

Participants: Sandrine Blazy, Vincent Laporte, David Pichardie.

We designed and proved sound, using the Coq proof assistant, a static analyzer, Verasco [26], based on abstract interpretation for 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, both relational and non-relational. Verasco integrates with the CompCert formally-verified C compiler so that not only the soundness of the analysis results is guaranteed with mathematical certitude, but also the fact that these guarantees carry over to the compiled code.

6.2.3. Certified Analyses for binary codes

Participants: Sandrine Blazy, Vincent Laporte, David Pichardie.

Static analysis of binary code is challenging for several reasons. In particular, standard static analysis techniques operate over control flow graphs, which are not available when dealing with self-modifying programs which can modify their own code at runtime. We formalized in the Coq proof assistant some key abstract interpretation techniques that automatically extract memory safety properties and control flow graphs from binary code [13], and operate over a small subset of the x86 assembly. Our analyzer is formally proved correct and has been run on several self-modifying challenges, provided by Cai et al. in their PLDI 2007 paper. This an extended version of out ITP 2014 paper.

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

Participants: Thomas Genet, Yann Salmon.

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. The tools we develop use, so-called, Tree Automata Completion 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. Now, we aim at applying this technique to the static analysis of programming languages whose semantics is based on terms, like functional programming languages. We already shown that static analysis of first order functional programs with a call-by-value evaluation strategy can be automated using tree automata completion [22]. This is the subject of the PhD thesis Yann Salmon has defended [11]. Now, one of the objective is to lift those results to the static analysis of higher-order functions.

6.4. Static analysis of functional specifications

Participants: Thomas Jensen, Oana Andreescu.

We have developed a static dependency analysis for a strongly typed, high-level functional specifications written in a 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, which only really impacts few invariants at the same time. Identifying those invariants that are unaffected by an operation can substantially ease the proof burden for the programmer. Our dependency analysis computes a conservative approximation of the input fragments on which the operations depend. It is a flow-sensitive interprocedural analysis that handles arrays, structures and variant data types. We have validated the scalability of the analysis to complex transition systems by applying it to a functional specification of the MINIX operating system. This work was reported in [25].

6.5. Semantics

6.5.1. Energy-valued semantics

Participant: David Cachera.

We develop a *-continuous Kleene ω-algebra of real-time energy functions [36]. Together with corresponding automata, these can be used to model systems which can consume and regain energy (or other types of resources) depending on available time. Using recent results on *-continuous Kleene ω-algebras and computability of certain manipulations on real-time energy functions, it follows that reachability and Büchi acceptance in real-time energy automata can be decided in a static way which only involves manipulations of real-time energy functions. This works opens the way to static analysis techniques for energy-valued semantics of real-time systems.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. The ANR VERASCO project

Participants: Sandrine Blazy, Delphine Demange, Vincent Laporte, David Pichardie.

Static program analysis, Certified static analysis

The VERASCO project (2012–2015) 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.

7.1.2. The ANR AnaStaSec project

Participants: Frédéric Besson, Sandrine Blazy, Thomas Jensen.

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.

7.1.3. The ANR Binsec project

Participants: Frédéric Besson, Sandrine Blazy, Pierre Wilke.

Binary code, Static program analysis

The Binsec project (2013–2017) is founded 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.

7.1.4. 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 (Estasys and Celtique), Inria Saclay (MAXPLUS) and VISEO/Object Direct.

7.1.5. 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.

7.1.6. 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.

7.1.7. 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.

7.2. International Initiatives

7.2.1. Inria Associate Teams not involved in an Inria International Labs

7.2.1.1. JCERT

Title: Verified Compilation of Concurrent Managed Languages

International Partner (Institution - Laboratory - Researcher):

Purdue University (United States) - Suresh Jagannathan

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#.

7.2.2. Inria International Partners

7.2.2.1. Declared Inria International Partners

Professor Philippa Gardner, Imperial College, UK, since December 2015.

7.2.2.2. Informal International Partners

Alan Schmitt is part of a Polonium Hubert Curien Partnership (PHC) with the University of Wrocław. This partnership is lead by Sergueï Lenglet, from Loria, Nancy, France.

7.3. International Research Visitors

7.3.1. Visits to International Teams

7.3.1.1. Sabbatical programme

Jensen Thomas

Date: Sep 2014 - Aug 2015

Institution: University of Copenhagen (Denmark)

7.3.1.2. Research stays abroad

Martin Bodin visited the Department of Computing at Imperial College London for three months.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

Static Analysis Symposium 2015, Saint-Malo, September 2015.

8.1.2. Scientific events selection

8.1.2.1. Chair of conference program committees

SAS 2015 (Static Analysis Symposium) was chaired by Sandrine Blazy and Thomas Jensen

8.1.2.2. Member of the conference program committees

- FORTE 2015 (International Conference on Formal Techniques for Distributed Objects, Components and Systems): Alan Schmitt
- CoqPL 2016 (International Workshop on Coq for PL): Alan Schmitt
- POPL 2015 (Symposium on Principles of Programming Languages): David Pichardie (Program Committee) and Delphine Demange (External Review Committee)
- ESOP 2015 (European Symposium on Programming) : Delphine Demange
- JFLA 2015 (Journées Francophones des Langages Applicatifs) : Delphine Demange
- OCaml workshop 2015: Sandrine Blazy
- AFADL 2015 (Approches Formelles dans l'Assistance au Développement de Logiciels) : Sandrine Blazy
- ITP 2015 (Interactive Theorem Proving): Sandrine Blazy
- CPP 2016 (Certified Proofs and Programs): Sandrine Blazy
- IFIP SEC 2015: Thomas Jensen

8.1.2.3. Reviewer

- CONCUR 2015 (International Conference on Concurrency Theory): Alan Schmitt
- FOSSACS 2016 (International Conference on Foundations of Software Science and Computation Structures): Alan Schmitt
- ITP 2015 (International Conference on Interactive Theorem Proving) : Delphine Demange
- ATVA 2015 (13th International Symposium on Automated Technology for Verification and Analysis) : Sandrine Blazy
- ESOP 2015 (European Symposium on Programming): Thomas Jensen.

8.1.3. Journal

8.1.3.1. Reviewer - Reviewing activities

- Applied Mathematics & Information Sciences: Alan Schmitt
- Formal Aspects of Computing: Alan Schmitt, Thomas Jensen
- International Journal of Computer Mathematics: Alan Schmitt
- Journal of Automated Reasoning: Sandrine Blazy, David Pichardie
- Journal of Computer Security : Sandrine Blazy

8.1.4. Invited talks

- Sandrine Blazy: Formal verification of compilers and static analyzers, PLMW@POPL 2015 (Programming Languages Mentoring Workshop)
- Thomas Jensen: Integrating formal verification within programming languages. Seminar Collège de France, October 2015.

8.1.5. Scientific expertise

- Sandrine Blazy: expertise of 2 projects for FNR (Luxemburg)
- Thomas Jensen, expertise of ANR AAP general, short proposals

8.1.6. Research administration

- Thomas Jensen is head of the NUMERIC department at University of Bretagne-Loire
- Thomas Jensen is managing the “Security” track of ANR Comin Labs.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

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

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

Licence: Delphine Demange, Functional Programming, 70h, L1, Université de Rennes 1 / Istic, France

Licence : Alan Schmitt, Programmation Fonctionnelle, 39h, L3, Insa Rennes, France

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

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

Licence : Sandrine Blazy, Functional programming, 30h, L3, 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 Cachera, Semantics of Programming Languages, 36h, M1, Université Rennes 1, France

Master : Frédéric Besson, Compilation, 50h, M1, Insa Rennes, France

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

Master : Alan Schmitt, Méthodes Formelles pour le développement de logiciels sûrs, 25h, M1, Rennes 1, 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, Software vulnerabilities, 38h, M2, Université Rennes 1, France

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

Master : Thomas Jensen, Program Analysis and Software Security, 39h, M2, Université Rennes 1, France

8.2.2. Supervision

PhD in progress : Martin Bodin, Certified Analyses of JavaScript, 1st september 2012, Thomas Jensen and Alan Schmitt

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

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

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

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

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: Pierre Wilke, Low-level memory models for compilers and static analysers, 1st august 2013, Sandrine Blazy and Frédéric Besson

PhD in progress: Oana Andreescu, Static analysis and automated program proving, 1st September 2013, Thomas Jensen

PhD in progress: Pauline Bolignano, Modeling and abstraction of system software, 1st November 2013, Thomas Jensen

PhD: Yann Salmon, Reachability for Term Rewriting Systems under Strategies, defended Dec 2015, Thomas Genet

PhD: Vincent Laporte, Verified static analyses for low-level languages, defended Nov 2015, Sandrine Blazy and David Pichardie

PhD: Stéphanie Riaud, Data obfuscation for protecting programs against dynamic analysis, defended Dec 2015, Sandrine Blazy

8.2.3. Juries

Alan Schmitt, jury member (reviewer) for the PhD defense of Burak Ekici, December 2015, Université Grenoble Alpes

David Pichardie, jury member (chair) for the PhD defense of Mounir Assaf, May 2015, University Rennes 1, France.

David Pichardie, jury member (reviewer) for the PhD defense of Bogdan Mihaila, January 2015, Technische Universität München, Germany.

Sandrine Blazy, jury member for the PhD defense of Alexis Fouilhé, October 2015, Grenoble University, France

Sandrine Blazy, jury member for the PhD defense of Zakaria Chihani, November 2015, Ecole Polytechnique, France.

Sandrine Blazy, jury member for the PhD defense of Lourdes del Carmen Gonzalez Huesca, November2015, Paris Diderot University, France.

Thomas Jensen, jury member for the PhD defense of Magnus Madsen, May 2015, Aarhus University, Denmark.

Thomas Jensen, jury member for the PhD defense of Clémentine Maurice, EURECOM, France.

Alan Schmitt, jury member for the selection of Inria CR (researcher) candidates, March and April 2015, Inria, Rennes, France.

Delphine Demange, jury member for the selection of a Maître de Conférences at University Paris Diderot (Paris 7) / LIAFA+PPS, May 2015, Paris, France.

Sandrine Blazy, jury member for the selection of a Maître de Conférences at University of Nancy, May 2015, Nancy, France.

Sandrine Blazy, jury member for the selection of a Maître de Conférences at University of Rennes 1, May 2015, Rennes, France.

Sandrine Blazy, jury member for the selection of a professor at University of Brest, May 2015, Brest, France.

Delphine Demange, jury member of the Gilles Kahn PhD award committee, December 2015, Inria Paris - Rocquencourt

8.3. Popularization

- IRISA 40th anniversary Open House - Recreational workshop (interactive demonstration) on the problem of functional correctness of programs, its importance in the field of critical software, and formal proof of correctness : Thomas Genet, Delphine Demange, Pauline Bolignano, Yannick Zakowski. Inria Convention centre, Inria Rennes - Bretagne Atlantique. Rennes, France. December 2015.
- Talk “Bug, Virus, Intrusion, Pirates... So many threats and no defense? Yes... maths.”, Thomas Genet, given two times in high schools close to Rennes.
- Two short tutorials. One for Isabelle/HOL [41] and one for SPAN+AVISPA [42].

9. Bibliography

Major publications by the team in recent years

[1] 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

[2] 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

[3] 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

[4] 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

[5] D. CACHERA, T. JENSEN, A. JOBIN, P. SOTIN. *Long-Run Cost Analysis by Approximation of Linear Operators over Dioids*, in "Mathematical Structures in Computer Science", 2010, vol. 20, n^{o} 4, pp. 589-624

[6] 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

[7] 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

[8] J. N. FOSTER, M. B. GREENWALD, J. T. MOORE, B. C. PIERCE, A. SCHMITT. *Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem*, in "Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages", Long Beach, CA, USA, 2005, pp. 233–246, Most Influential POPL Paper Award, 2015. Extended version available as University of Pennsylvania technical report MS-CIS-03-08. Earlier version presented at the Workshop on Programming Language Technologies for XML (PLAN-X), 2004

[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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] Y. SALMON. *Reachability Analysis for Functional Programs with call-by-value evaluation strategy*, Université de Rennes 1, December 2015, https://hal.inria.fr/tel-01250252

Articles in International Peer-Reviewed Journals

[12] S. BLAZY, D. BÜHLER, B. YAKOBOWSKI. *Improving static analyses of C programs with conditional predicates*, in "Science of Computer Programming", January 2016, Accepted manuscript. Available online 30 November 2015.Extended version of the FMICS 2014 paper, https://hal.inria.fr/hal-01242077

[13] S. BLAZY, V. LAPORTE, D. PICHARDIE. *Verified Abstract Interpretation Techniques for Disassembling Low-level Self-modifying Code*, in "Journal of Automated Reasoning", 2016, Version étendue de l'article de la conférence ITP 2014, accepté pour publication, https://hal.inria.fr/hal-01243700

[14] 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

[15] P. GENEVÈS, N. LAYAÏDA, A. SCHMITT, N. GESBERT. *Efficiently Deciding μ-calculus with Converse over Finite Trees*, in "ACM Transactions on Computational Logic", March 2015, vol. 16, n^o 2, 41 p. [*DOI :* 10.1145/2724712], https://hal.inria.fr/hal-00868722

[16] B. MARINKOVIC, V. CIANCAGLINI, Z. OGNJANOVIC, P. GLAVAN, L. LIQUORI, P. MAKSIMOVIC. *Analyzing the exhaustiveness of the synapse protocol*, in "Peer-to-Peer Networking and Applications, Springer", July 2015, vol. 8, n^o 5, pp. 793–806 [*DOI :* 10.1007/S12083-014-0293-Z], https://hal.inria.fr/hal-01146050

International Conferences with Proceedings

[17] S. BLAZY, A. MARONEZE, D. PICHARDIE. *Verified Validation of Program Slicing*, in "CPP 2015 : Conference on Certified Programs and Proofs", Mumbai, India, 2015, pp. 109-117 [*DOI :* 10.1145/2676724.2693169], https://hal.inria.fr/hal-01110821

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

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

[20] D. DEMANGE, D. PICHARDIE, L. STEFANESCO. *Verifying Fast and Sparse SSA-based Optimizations in Coq*, in "24th International Conference on Compiler Construction, CC 2015", London, United Kingdom, 2015, https://hal.inria.fr/hal-01110779

[21] J. FRAGOSO SANTOS, T. JENSEN, T. REZK, A. SCHMITT. *Hybrid Typing of Secure Information Flow in a JavaScript-like Language*, in "International Symposium on Trustworthy Global Computing - (TGC 2015)", Madrid, Spain, August 2015, https://hal.archives-ouvertes.fr/hal-01243029

[22] T. GENET, Y. SALMON. *Reachability Analysis of Innermost Rewriting*, in "RTA", Warshaw, Poland, 2015, pp. 1-17 [*DOI :* 10.4230/LIPICS.RTA.2015.X], https://hal.inria.fr/hal-01194530

[23] P. GENEVÈS, A. SCHMITT. *Expressive Logical Combinators for Free*, in "International Joint Conference on Artificial Intelligence (IJCAI 2015)", Buenos Aires, Argentina, July 2015, https://hal.inria.fr/hal-00868724

[24] F. HONSELL, L. LIQUORI, P. MAKSIMOVIC, I. SCAGNETTO. *Gluing together Proof Environments: Canonical extensions of LF Type Theories featuring Locks* , in "LFMTP'15. 9th International Workshop on Logical Frameworks and Meta-languages, Berlin, Germany", Berlin, Germany, August 2015, vol. Electronic Proceedings in Theoretical Computer Science (EPTCS) [*DOI :* 10.4204/EPTCS.185.1], https://hal.archives-ouvertes.fr/hal-01170029

[25] T. JENSEN, S. LESCUYER, A. OANA. *Dependency Analysis of Functional Specifications with Algebraic Data Structures*, in "17th International Conference on Formal Engineering Methods, ICFEM 2015", Paris, France, Springer LNCS, Springer Verlag, November 2015, vol. 9407, 18 p. [*DOI :* 10.1007/978-3-319-25423-4_8], https://hal.inria.fr/hal-01243002

[26] 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

[27] B. KORDY, R. JHAWAR, S. MAUW, S. RADOMIROVIC, R. TRUJILLO-RASUA. *Attack Trees with Sequential Conjunction*, in "IFIP SEC 2015 - International Conference on ICT Systems Security and Privacy Protection", Hamburg, Germany, H. FEDERRATH, D. GOLLMANN (editors), Springer, May 2015, vol. IFIP Advances in Information and Communication Technology, n^{o} 455, pp. 339-353 [*DOI :* 10.1007/978-3-319-18467-8_23], https://hal.inria.fr/hal-01197256

[28] P. MAKSIMOVIC, A. SCHMITT. *HOCore in Coq*, in "The 6th conference on Interactive Theorem Proving - (ITP 2015)", Nanjing, China, Springer, August 2015, vol. 9236 [*DOI :* 10.1007/978-3-319-22102-1_19], https://hal.archives-ouvertes.fr/hal-01243017

National Conferences with Proceedings

[29] M. ESCARRÁ, M. PETAR, A. SCHMITT. *HOCore in Coq*, in "Vingt-sixièmes Journées Francophones des Langages Applicatifs (JFLA 2015)", Le Val d'Ajol, France, D. BAELDE, J. ALGLAVE (editors), January 2015, https://hal.inria.fr/hal-01099130

Conferences without Proceedings

[30] R. ANDRIATSIMANDEFITRA RATSISAHANANA, T. GENET, L. GUILLO, J.-F. LALANDE, D. PICHARDIE, V. VIET TRIEM TONG. *Kharon : Découvrir, comprendre et reconnaître des malware Android par suivi de flux d'information*, in "Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information", Troyes, France, May 2015, https://hal.inria.fr/hal-01154368

[31] A. BART, C. TRUCHET, E. MONFROY. *Verifying a Real-Time Language with Constraints*, in "27th IEEE International Conference on Tools with Artificial Intelligence", Vietri sul Mare, Italy, 2015, https://hal.archives-ouvertes.fr/hal-01234188

[32] F. BESSON, S. BLAZY, P. WILKE. *A Concrete Memory Model for CompCert*, in "ITP 2015 : 6th International Conference on Interactive Theorem Proving", Nanjing, China, SPRINGER (editor), August 2015, vol. Lecture Notes in Computer Science (LNCS), n^o 9236, pp. 67-83 [*DOI :* 10.1007/978-3-319-22102-1_5], https://hal.inria.fr/hal-01194549

[33] S. BLAZY. *Formal verification of compilers and static analyzers.* , in "PLMW@POPL 2015 - Programming Languages Mentoring Workshop", Mumbai, India, January 2015, https://hal.inria.fr/hal-01242094

[34] S. BLAZY, D. DEMANGE, D. PICHARDIE. *Validating Dominator Trees for a Fast, Verified Dominance Test*, in "Interactive Theorem Proving", Nanjing, China, SPRINGER (editor), August 2015, vol. Lecture Notes in Computer Science (LNCS), n^o 9236 [*DOI :* 10.1007/978-3-319-22102-1_6], https://hal.inria.fr/hal-01193281

[35] S. BLAZY, S. RIAUD, T. SIRVENT. *Data Tainting and Obfuscation: Improving Plausibility of Incorrect Taint*, in "Source Code Analysis and Manipulation (SCAM)", Bremen, Germany, IEEE (editor), September 2015, https://hal.inria.fr/hal-01193286

[36] D. CACHERA, U. FAHRENBERG, A. LEGAY. *An ω-Algebra for Real-Time Energy Problems*, in "35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science", Bengaluru, India, December 2015, https://hal.inria.fr/hal-01237667

[37] T. GENET, B. KORDY, A. VANSYNGEL. *Vers un outil de vérification formelle légere pour OCaml*, in "AFADL", Bordeaux, France, 2015, 6 p. , https://hal.inria.fr/hal-01194538

[38] S. LENGLET, A. SCHMITT. *Howe's Method for Contextual Semantics* , in "CONCUR 2015 26th International Conference on Concurrency Theory", Madrid, Spain, September 2015 [*DOI :* 10.4230/LIPICS.CONCUR.2015.212], https://hal.inria.fr/hal-01192699

[39] 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

Scientific Books (or Scientific Book chapters)

[40] T. J. SANDRINE BLAZY (editor). *Static Analysis Symposium - 22nd International Symposium, SAS 2015, Saint-Malo, France, September 9-11, 2015. Proceedings*, Springer, Saint-Malo, France, August 2015, vol. Lecture Notes in Computer Science (LNCS), n^{o} 9291, 335 p. , https://hal.inria.fr/hal-01194558

Research Reports

[41] T. GENET. *A Short Isabelle/HOL Tutorial for the Functional Programmer*, IRISA, 2015, https://hal.inria.fr/hal-01208577

[42] T. GENET. *A Short SPAN+AVISPA Tutorial*, IRISA, October 2015, https://hal.inria.fr/hal-01213074

[43] S. LENGLET, A. SCHMITT. *Howe's Method for Contextual Semantics*, Inria, June 2015, n^{o} RR-8750, 31 p. , https://hal.inria.fr/hal-01168865

Other Publications

[44] F. HONSELL, L. LIQUORI, P. MAKSIMOVIC, I. SCAGNETTO. *LLFP : A Logical Framework for modeling External Evidence, Side Conditions, and Proof Irrelevance using Monads*, January 2015, working paper or preprint [*DOI :* 10.4204/EPTCS.185.1], https://hal.inria.fr/hal-01146059

References in notes

[45] *The Coq Proof Assistant*, 2009, http://coq.inria.fr/

[46] E. ALBERT, P. ARENAS, S. GENAIM, G. PUEBLA, D. ZANARDINI. *COSTA: Design and Implementation of a Cost and Termination Analyzer for Java Bytecode*, in "FMCO", 2007, pp. 113-132

[47] D. ASPINALL, L. BERINGER, M. HOFMANN, HANS-WOLFGANG. LOIDL, A. MOMIGLIANO. *A Program Logic for Resource Verification*, in "In Proceedings of the 17th International Conference on Theorem Proving in Higher-Order Logics, (TPHOLs 2004), volume 3223 of LNCS", Springer, 2004, pp. 34–49

[48] D. F. BACON, P. F. SWEENEY. *Fast Static Analysis of C++ Virtual Function Calls*, in "OOPSLA'96", 1996, pp. 324-341

[49] P. BAILLOT, P. COPPOLA, U. D. LAGO. *Light Logics and Optimal Reduction: Completeness and Complexity*, in "LICS", 2007, pp. 421-430

[50] E. BALLAND, Y. BOICHUT, T. GENET, P.-E. MOREAU. *Towards an Efficient Implementation of Tree Automata Completion*, in "Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008", Lectures Notes in Computer Science, Springer-Verlag, 2008, vol. 5140, pp. 67-82

[51] 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

[52] F. BESSON, S. BLAZY, P. WILKE. *A Precise and Abstract Memory Model for C Using Symbolic Values*, in "12th Asian Symposium on Programming Languages and Systems (APLAS 2014)", Singapore, Singapore, LNCS, Springer, 2014, vol. 8858, pp. 449 - 468 [*DOI :* 10.1007/978-3-319-12736-1_24], https://hal.inria.fr/hal-01093312

[53] F. BESSON, T. JENSEN. *Modular Class Analysis with DATALOG*, in "SAS'2003", 2003, pp. 19-36

[54] D. CACHERA, T. JENSEN, A. JOBIN, P. SOTIN. *Long-Run Cost Analysis by Approximation of Linear Operators over Dioids*, in "Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008", Lectures Notes in Computer Science, Springer-Verlag, 2008, vol. 5140, pp. 122-138

[55] 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

[56] D. CACHERA, T. JENSEN, D. PICHARDIE, G. SCHNEIDER. *Certified Memory Usage Analysis*, in "Proc. of 13th International Symposium on Formal Methods (FM'05)", LNCS, Springer-Verlag, 2005

[57] P. COUSOT, R. COUSOT. *Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*, in "Proc. of POPL'77", 1977, pp. 238–252

[58] A. ERMEDAHL, C. SANDBERG, J. GUSTAFSSON, S. BYGDE, B. LISPER. *Loop Bound Analysis based on a Combination of Program Slicing, Abstract Interpretation, and Invariant Analysis*, in "Seventh International Workshop on Worst-Case Execution Time Analysis, (WCET'2007)", July 2007, http://www.mrtc.mdh.se/index.php?choice=publications&id=1317

[59] G. FEUILLADE, T. GENET, V. VIET TRIEM TONG. *Reachability Analysis over Term Rewriting Systems*, in "Journal of Automated Reasoning", 2004, vol. 33, n^{o} 3–4, pp. 341–383

[60] M. FÄHNDRICH, K. R. M. LEINO. *Declaring and checking non-null types in an object-oriented language*, in "OOPSLA", 2003, pp. 302-312

[61] T. GENET. *Decidable Approximations of Sets of Descendants and Sets of Normal forms*, in "RTA'98", LNCS, Springer, 1998, vol. 1379, pp. 151–165

[62] T. GENET, V. VIET TRIEM TONG. *Reachability Analysis of Term Rewriting Systems with Timbuk*, in "LPAR'01", LNAI, Springer, 2001, vol. 2250, pp. 691-702

[63] D. GROVE, C. CHAMBERS. *A framework for call graph construction algorithms*, in "Toplas", 2001, vol. 23, n^{o} 6, pp. 685–746

[64] D. GROVE, G. DEFOUW, J. DEAN, C. CHAMBERS. *Call graph construction in object-oriented languages*, in "ACM SIGPLAN Notices", 1997, vol. 32, n^{o} 10, pp. 108–124

[65] M. HOFMANN, S. JOST. *Static prediction of heap space usage for first-order functional programs*, in "POPL", 2003, pp. 185-197

[66] L. HUBERT. *A Non-Null annotation inferencer for Java bytecode*, in "Proc. of the Workshop on Program Analysis for Software Tools and Engineering (PASTE'08)", ACM, 2008

[67] L. HUBERT, T. JENSEN, D. PICHARDIE. *Semantic foundations and inference of non-null annotations*, in "Proc. of the 10th International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'08)", Lecture Notes in Computer Science, Springer-Verlag, 2008, vol. 5051, pp. 132-149

[68] X. LEROY. *A formally verified compiler back-end*, in "Journal of Automated Reasoning", December 2009, vol. 43, n^{o} 4, pp. 363-446 [*DOI :* 10.1007/s10817-009-9155-4], https://hal.inria.fr/inria-00360768

[69] O. LHOTÁK, L. J. HENDREN. *Evaluating the benefits of context-sensitive points-to analysis using a BDD-based implementation*, in "ACM Trans. Softw. Eng. Methodol.", 2008, vol. 18, n^{o} 1

[70] V. B. LIVSHITS, M. S. LAM. *Finding Security Errors in Java Programs with Static Analysis*, in "Proc. of the 14th Usenix Security Symposium", 2005, pp. 271–286

[71] A. MILANOVA, A. ROUNTEV, B. G. RYDER. *Parameterized object sensitivity for points-to analysis for Java*, in "ACM Trans. Softw. Eng. Methodol.", 2005, vol. 14, n^{o} 1, pp. 1–41

[72] M. NAIK, A. AIKEN. *Conditional must not aliasing for static race detection*, in "POPL'07", ACM, 2007, pp. 327-338

[73] M. NAIK, A. AIKEN, J. WHALEY. *Effective static race detection for Java*, in "PLDI'2006", ACM, 2006, pp. 308-319

[74] F. NIELSON, H. NIELSON, C. HANKIN. *Principles of Program Analysis*, Springer, 1999

[75] J. PALSBERG, M. SCHWARTZBACH. *Object-Oriented Type Inference*, in "OOPSLA'91", 1991, pp. 146-161

[76] J. PALSBERG, M. SCHWARTZBACH. *Object-Oriented Type Systems*, John Wiley & Sons, 1994

[77] D. PICHARDIE. *Interprétation abstraite en logique intuitionniste : extraction d'analyseurs Java certiés*, Université Rennes 1, Rennes, France, dec 2005

[78] A. D. PIERRO, H. WIKLICKY. *Operator Algebras and the Operational Semantics of Probabilistic Languages*, in "Electr. Notes Theor. Comput. Sci.", 2006, vol. 161, pp. 131-150

[79] P. SOTIN, D. CACHERA, T. JENSEN. *Quantitative Static Analysis over semirings: analysing cache behaviour for Java Card*, in "4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006)", Electronic Notes in Theoretical Computer Science, Elsevier, 2006, vol. 164, pp. 153-167

[80] F. TIP, J. PALSBERG. *Scalable propagation-based call graph construction algorithms*, in "OOPSLA", 2000, pp. 281-293

[81] J. WHALEY, M. S. LAM. *Cloning-based context-sensitive pointer alias analysis using binary decision diagrams*, in "PLDI '04", ACM, 2004, pp. 131–144