IN PARTNERSHIP WITH:
Université Paris-Sud (Paris 11)

Activity Report 2016

Project-Team TOCCATA

Formally Verified Programs, Certified Tools, Numerical Computations

IN COLLABORATION WITH: Laboratoire de recherche en informatique (LRI)

RESEARCH CENTER
Saclay - Île-de-France

THEME
Proofs and Verification

Table of contents

Project-Team TOCCATA

Creation of the Team: 2012 September 01, updated into Project-Team: 2014 July 01

Keywords:

Computer Science and Digital Science:

2.1.3. - Functional programming
2.1.6. - Concurrent programming
2.1.11. - Proof languages
2.4.2. - Model-checking
2.4.3. - Proofs
7.4. - Logic in Computer Science
7.12. - Computer arithmetic

Other Research Topics and Application Domains:

5.2.2. - Railway
5.2.3. - Aviation
5.2.4. - Aerospace

1. Members

Research Scientists

Claude Marché [Inria, Senior Researcher, Team leader, HDR]
Sylvie Boldo [Inria, Researcher, Team co-leader, HDR]
Arthur Charguéraud [Inria, Researcher, until Sep. 2016]
Jean-Christophe Filliâtre [CNRS, Senior Researcher, HDR]
Guillaume Melquiond [Inria, Researcher]

Faculty Members

Sylvain Conchon [Univ. Paris-Sud, Professor, HDR]
Andrei Paskevich [Univ. Paris-Sud, Associate Professor]
Christine Paulin-Mohring [Univ. Paris-sud, Professor, until Jun. 2016, HDR]

Engineers

Clément Fumex [Inria]
Sylvain Dailler [Inria, from Jun. 2016]

PhD Students

Cláudio Belo Da Silva Lourenço [Universidade do Minho, Portugal, visiting from Jun. to Dec. 2016]
Ran Chen [Institute of Software, Chinese Academy of Sciences, Beijing, China, visiting from May 2016 to Feb. 2017]
Martin Clochard [Univ. Paris-Sud (ENS grant)]
Albin Coquereau [ENSTA]
David Declerck [Univ. Paris-Sud]
Florian Faissole [Inria, from Oct 2016]
Léon Gondelman [Inria]
Mario José Parreira Pereira [Grant Portuguese government]
Mattias Roux [Univ. Paris-Sud]

Administrative Assistants

Régine Bricquet [Inria, until Jan 2016]

Katia Evrat [Inria, from Feb 2016]

Others

Thibaut Balabonski [Univ. Paris-Sud, associate member]
Chantal Keller [Univ. Paris-Sud, associate member]
Raphaël Rieu-Helft [Visitor, ENS, from Sep 2016]
Xavier Urbain [ENSIEE & LRI, associate member, until Aug 2016]

2. Overall Objectives

2.1. Overall Objectives

The general objective of the Toccata project is to promote formal specification and computer-assisted proof in the development of software that requires high assurance in terms of safety and correctness with respect to the intended behavior of the software.

2.1.1. Context

The importance of software in critical systems increased a lot in the last decade. Critical software appears in various application domains like transportation (e.g., aviation, railway), communication (e.g., smartphones), banking, etc. The number of tasks performed by software is quickly increasing, together with the number of lines of code involved. Given the need of high assurance of safety in the functional behavior of such applications, the need for automated (i.e., computer-assisted) methods and techniques to bring guarantee of safety became a major challenge. In the past and at present, the most widely used approach to check safety of software is to apply heavy test campaigns. These campaigns take a large part of the costs of software development, yet they cannot ensure that all the bugs are caught.

Generally speaking, software verification approaches pursue three goals: (1) verification should be sound, in the sense that no bugs should be missed, (2) verification should not produce false alarms, or as few as possible (3) it should be as automated as possible. Reaching all three goals at the same time is a challenge. A large class of approaches emphasizes goals (2) and (3): testing, run-time verification, symbolic execution, model checking, etc. Static analysis, such as abstract interpretation, emphasizes goals (1) and (3). Deductive verification emphasizes (1) and (2). The Toccata project is mainly interested in exploring the deductive verification approach, although we also consider the others in some cases.

In the past decade, there has been significant progress made in the domain of deductive program verification. They are emphasized by some success stories of application of these techniques on industrial-scale software. For example, the *Atelier B* system was used to develop part of the embedded software of the Paris metro line 14 [45] and other railroad-related systems; a formally proved C compiler was developed using the Coq proof assistant [100]; Microsoft's hypervisor for highly secure virtualization was verified using VCC [79] and the Z3 prover [119]; the L4-verified project developed a formally verified micro-kernel with high security guarantees, using analysis tools on top of the Isabelle/HOL proof assistant [96]. Another sign of recent progress is the emergence of deductive verification competitions (e.g., VerifyThis [3], VScomp [90]).

Finally, recent trends in the industrial practice for development of critical software is to require more and more guarantees of safety, e.g., the upcoming DO-178C standard for developing avionics software adds to the former DO-178B the use of formal models and formal methods. It also emphasizes the need for certification of the analysis tools involved in the process.

2.1.2. Deductive verification

There are two main families of approaches for deductive verification. Methods in the first family build on top of mathematical proof assistants (e.g., Coq, Isabelle) in which both the model and the program are encoded; the proof that the program meets its specification is typically conducted in an interactive way using the underlying proof construction engine. Methods from the second family proceed by the design of standalone tools taking as input a program in a particular programming language (e.g., C, Java) specified with a dedicated annotation

language (e.g., ACSL [44], JML [64]) and automatically producing a set of mathematical formulas (the *verification conditions*) which are typically proved using automatic provers (e.g., Z3, Alt-Ergo [46], CVC3 [43], CVC4).

The first family of approaches usually offers a higher level of assurance than the second, but also demands more work to perform the proofs (because of their interactive nature) and makes them less easy to adopt by industry. Moreover, they do not allow to directly analyze a program written in a mainstream programming language like Java or C. The second kind of approaches has benefited in the past years from the tremendous progress made in SAT and SMT solving techniques, allowing more impact on industrial practices, but suffers from a lower level of trust: in all parts of the proof chain (the model of the input programming language, the VC generator, the back-end automatic prover), potential errors may appear, compromising the guarantee offered. Moreover, while these approaches are applied to mainstream languages, they usually support only a subset of their features.

3. Research Program

3.1. Introduction

In the former ProVal project, we have been working on the design of methods and tools for deductive verification of programs. One of our original skills was the ability to conduct proofs by using automatic provers and proof assistants at the same time, depending on the difficulty of the program, and specifically the difficulty of each particular verification condition. We thus believe that we are in a good position to propose a bridge between the two families of approaches of deductive verification presented above. Establishing this bridge is one of the goals of the Toccata project: we want to provide methods and tools for deductive program verification that can offer both a high amount of proof automation and a high guarantee of validity. Toward this objective, a new axis of research was proposed: the development of *certified* analysis tools that are themselves formally proved correct.

The reader should be aware that the word “certified” in this scientific programme means “verified by a formal specification and a formal proof that the program meets this specification”. This differs from the standard meaning of “certified” in an industrial context where it means a conformance to some rigorous process and/or norm. We believe this is the right term to use, as it was used for the *Certified Compiler* project [100], the new conference series *Certified Programs and Proofs*, and more generally the important topics of *proof certificates*.

In industrial applications, numerical calculations are very common (e.g. control software in transportation). Typically they involve floating-point numbers. Some of the members of Toccata have an internationally recognized expertise on deductive program verification involving floating-point computations. Our past work includes a new approach for proving behavioral properties of numerical C programs using Frama-C/Jessie [41], various examples of applications of that approach [61], the use of the Gappa solver for proving numerical algorithms [118], an approach to take architectures and compilers into account when dealing with floating-point programs [62], [111]. We also contributed to the Handbook of Floating-Point Arithmetic [110]. A representative case study is the analysis and the proof of both the method error and the rounding error of a numerical analysis program solving the one-dimension acoustic wave equation [4] [54]. Our experience led us to a conclusion that verification of numerical programs can benefit a lot from combining automatic and interactive theorem proving [56], [61]. Certification of numerical programs is the other main axis of Toccata.

Our scientific programme in structured into four objectives:

1. deductive program verification;
2. automated reasoning;
3. formalization and certification of languages, tools and systems;
4. proof of numerical programs.

We detail these objectives below.

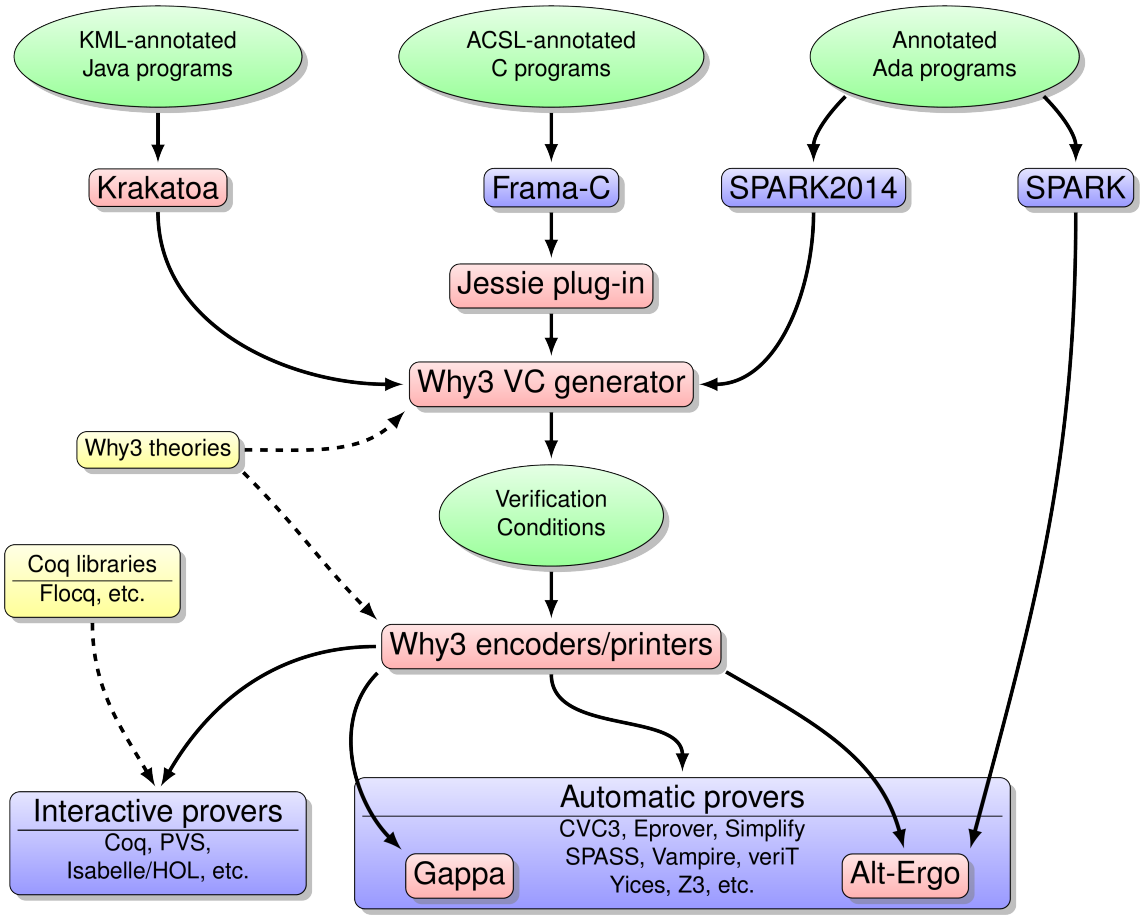

Figure 1. The Why3 ecosystem

3.2. Deductive Program Verification

Permanent researchers: A. Charguéraud, S. Conchon, J.-C. Filliâtre, C. Marché, G. Melquiond, A. Paskevich

3.2.1. The Why3 Ecosystem

This ecosystem is central in our work; it is displayed on Figure 1. The boxes in red background correspond to the tools we develop in the Toccata team.

- The initial design of Why3 was presented in 2012 [49], [89]. In the past years, the main improvements concern the specification language (such as support for higher-order logic functions [67]) and the support for provers. Several new interactive provers are now supported: PVS 6 (used at NASA), Isabelle2014 (planned to be used in the context of Ada program via Spark), and Mathematica. We also added support for new automated provers: CVC4, Metitarski, Metis, Beagle, Princess, and Yices2. More technical improvements are the design of a Coq tactic to call provers via Why3 from Coq, and the design of a proof session mechanism [48]. Why3 was presented during several invited talks [88], [87], [84], [85].
- At the level of the C front-end of Why3 (via Frama-C), we have proposed an approach to add a notion of refinement on C programs [117], and an approach to reason about pointer programs with a standard logic, via *separation predicates* [47]
- The Ada front-end of Why3 has mainly been developed during the past three years, leading to the release of SPARK2014 [95] (http://www.spark-2014.org/)
- In collaboration with J. Almeida, M. Barbosa, J. Pinto, and B. Vieira (University do Minho, Braga, Portugal), J.-C. Filliâtre has developed a method for certifying programs involving cryptographic methods. It uses Why as an intermediate language [40].
- With M. Pereira and S. Melo de Sousa (Universidade da Beira Interior, Covilhã, Portugal), J.-C. Filliâtre has developed an environment for proving ARM assembly code. It uses Why3 as an intermediate VC generator. It was presented at the Inforum conference [114] (best student paper).

3.2.2. Concurrent Programming

- S. Conchon and A. Mebsout, in collaboration with F. Zaïdi (VALS team, LRI), A. Goel and S. Krstić (Strategic Cad Labs, INTEL) have proposed a new model-checking approach for verifying safety properties of array-based systems. This is a syntactically restricted class of parametrized transition systems with states represented as arrays indexed by an arbitrary number of processes. Cache coherence protocols and mutual exclusion algorithms are typical examples of such systems. It was first presented at CAV 2012 [7] and detailed further [77]. It was applied to the verification of programs with fences [73]. The core algorithm has been extended with a mechanism for inferring invariants. This new algorithm, called BRAB, is able to automatically infer invariants strong enough to prove industrial cache coherence protocols. BRAB computes over-approximations of backward reachable states that are checked to be unreachable in a finite instance of the system. These approximations (candidate invariants) are then model-checked together with the original safety properties. Completeness of the approach is ensured by a mechanism for backtracking on spurious traces introduced by too coarse approximations [74], [106].
- In the context of the ERC DeepSea project [1], A. Charguéraud and his co-authors have developed a unifying semantics for various different paradigms of parallel computing (fork-join, async-finish, and futures), and published a conference paper describing this work [16]. Besides, A. Charguéraud and his co-authors have polished their previous work on granularity control for parallel algorithms using user-provided complexity functions, and produced a journal article [11].

[1] Arthur Charguéraud is involved 40% of his time in the ERC DeepSea project, which is hosted at Inria Paris Rocquencourt (team Gallium).

3.2.3. Case Studies

- To provide an easy access to the case studies that we develop using Why3 and its front-ends, we have published a *gallery of verified programs* on our web page http://toccata.lri.fr/gallery/. Part of these examples are the solutions to the competitions VerifyThis 2011 [63], VerifyThis 2012 [3], and the competition VScomp 2011 [90].
- Other case studies that led to publications are the design of a library of data-structures based on AVLs [66], and the verification a two-lines C program (solving the N-queens puzzle) using Why3 [86].
- A. Charguéraud, with F. Pottier (Inria Paris), extended their formalization of the correctness and asympotic complexity of the classic Union Find data structure, which features the bound expressed in terms of the inverse Ackermann function [39]. The proof, conducted using CFML extended with time credits, was refined using a slightly more complex potential function, allowing to derive a simpler and richer interface for the data structure. A journal article is in preparation.

For other case studies, see also sections of numerical programs and formalization of languages and tools.

3.2.4. Project-team Positioning

Several research groups in the world develop their own approaches, techniques, and tools for deductive verification. With respect to all these related approaches and tools, our originality is our will to use more sophisticated specification languages (with inductive definitions, higher-order features and such) and the ability to use a large set of various theorem provers, including the use of interactive theorem proving to deal with complex functional properties.

- The RiSE team [2] at Microsoft Research Redmond, USA, partly in collaboration with team "programming methodology" team [3] at ETH Zurich develop tools that are closely related to ours: Boogie and Dafny are direct competitors of Why3, VCC is a direct competitor of Frama-C/Jessie.
- The KeY project [4] (several teams, mainly at Karlsruhe and Darmstadt, Germany, and Göteborg, Sweden) develops the KeY tool for Java program verification [38], based on dynamic logic, and has several industrial users. They use a specific modal logic (dynamic logic) for modeling programs, whereas we use standard logic, so as to be able to use off-the-shelf automated provers.
- The "software engineering" group at Augsburg, Germany, develops the KIV system [5], which was created more than 20 years ago (1992) and is still well maintained and efficient. It provides a semi-interactive proof environment based on algebraic-style specifications, and is able to deal with several kinds of imperative style programs. They have a significant industrial impact.
- The VeriFast system [6] aims at verifying C programs specified in Separation Logic. It is developed at the Katholic University at Leuven, Belgium. We do not usually use separation logic (so as to use off-the-shelf provers) but alternative approaches (e.g. static memory separation analysis).
- The Mobius Program Verification Environment [7] is a joint effort for the verification of Java source annotated with JML, combining static analysis and runtime checking. The tool ESC/Java2 [8] is a VC generator similar to Krakatoa, that builds on top of Boogie. It is developed by a community leaded by University of Copenhagen, Denmark. Again, our specificity with respect to them is the consideration of more complex specification languages and interactive theorem proving.
- The Lab for Automated Reasoning and Analysis [9] at EPFL, develop methods and tools for verification of Java (Jahob) and Scala (Leon) programs. They share with us the will and the ability to use several provers at the same time.
- The TLA environment [10], developed by Microsoft Research and the Inria team Veridis, aims at

[2] http://research.microsoft.com/en-us/groups/rise/default.aspx
[3] http://www.pm.inf.ethz.ch/
[4] http://www.key-project.org/
[5] http://www.isse.uni-augsburg.de/en/software/kiv/
[6] http://people.cs.kuleuven.be/~bart.jacobs/verifast/
[7] http://kindsoftware.com/products/opensource/Mobius/
[8] http://kindsoftware.com/products/opensource/ESCJava2/
[9] http://lara.epfl.ch/w/

the verification of concurrent programs using mathematical specifications, model checking, and interactive or automated theorem proving.

- The F* project [11], developed by Microsoft Research and the Inria Prosecco team, aims at providing a rich environment for developing programs and proving them.

The KeY and KIV environments mentioned above are partly based on interactive theorem provers. There are other approaches on top of general-purpose proof assistants for proving programs that are not purely functional:

- The Ynot project [12] is a Coq library for writing imperative programs specified in separation logic. It was developed at Harvard University, until the end of the project in 2010. Ynot had similar goals as CFML, although Ynot requires programs to be written in monadic style inside Coq, whereas CFML applies directly on programs written in OCaml syntax, translating them into logical formulae.
- Front-ends to Isabelle were developed to deal with simple sequential imperative programs [116] or C programs [113]. The L4-verified project [96] is built on top of Isabelle.

3.3. Automated Reasoning

Permanent researchers: S. Conchon, G. Melquiond, A. Paskevich

3.3.1. Generalities on Automated Reasoning

- J. C. Blanchette and A. Paskevich have designed an extension to the TPTP TFF (Typed First-order Form) format of theorem proving problems to support rank-1 polymorphic types (also known as ML-style parametric polymorphism) [1]. This extension, named TFF1, has been incorporated in the TPTP standard.
- S. Conchon defended his *habilitation à diriger des recherches* in December 2012. The memoir [70] provides a useful survey of the scientific work of the past 10 years, around the SMT solving techniques, that led to the tools Alt-Ergo and Cubicle as they are nowadays.

3.3.2. Quantifiers and Triggers

- C. Dross, J. Kanig, S. Conchon, and A. Paskevich have proposed a generic framework for adding a decision procedure for a theory or a combination of theories to an SMT prover. This mechanism is based on the notion of instantiation patterns, or *triggers*, which restrict instantiation of universal premises and can effectively prevent a combinatorial explosion. A user provides an axiomatization with triggers, along with a proof of completeness and termination in the proposed framework, and obtains in return a sound, complete and terminating solver for his theory. A prototype implementation was realized on top of Alt-Ergo. As a case study, a feature-rich axiomatization of doubly-linked lists was proved complete and terminating [82]. C. Dross defended her PhD thesis in April 2014 [83]. The main results of the thesis are: (1) a formal semantics of the notion of *triggers* typically used to control quantifier instantiation in SMT solvers, (2) a general setting to show how a first-order axiomatization with triggers can be proved correct, complete, and terminating, and (3) an extended DPLL(T) algorithm to integrate a first-order axiomatization with triggers as a decision procedure for the theory it defines. Significant case studies were conducted on examples coming from SPARK programs, and on the benchmarks on B set theory constructed within the BWare project.

3.3.3. Reasoning Modulo Theories

- S. Conchon, É. Contejean and M. Iguernelala have presented a modular extension of ground AC-completion for deciding formulas in the combination of the theory of equality with user-defined AC symbols, uninterpreted symbols and an arbitrary signature-disjoint Shostak theory X [72]. This work extends the results presented in [71] by showing that a simple preprocessing step allows to get rid of a full AC-compatible reduction ordering, and to simply use a partial multiset extension of a *non-necessarily AC-compatible* ordering.

[10] http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html
[11] http://research.microsoft.com/en-us/projects/fstar/
[12] http://ynot.cs.harvard.edu/

- S. Conchon, M. Iguernelala, and A. Mebsout have designed a collaborative framework for reasoning modulo simple properties of non-linear arithmetic [76]. This framework has been implemented in the Alt-Ergo SMT solver.
- S. Conchon, G. Melquiond and C. Roux have described a dedicated procedure for a theory of floating-point numbers which allows reasoning on approximation errors. This procedure is based on the approach of the Gappa tool: it performs saturation of consequences of the axioms, in order to refine bounds on expressions. In addition to the original approach, bounds are further refined by a constraint solver for linear arithmetic [78]. This procedure has been implemented in Alt-Ergo.
- In collaboration with A. Mahboubi (Inria project-team Typical), and G. Melquiond, the group involved in the development of Alt-Ergo have implemented and proved the correctness of a novel decision procedure for quantifier-free linear integer arithmetic [2]. This algorithm tries to bridge the gap between projection and branching/cutting methods: it interleaves an exhaustive search for a model with bounds inference. These bounds are computed provided an oracle capable of finding constant positive linear combinations of affine forms. An efficient oracle based on the Simplex procedure has been designed. This algorithm is proved sound, complete, and terminating and is implemented in Alt-Ergo.
- Most of the results above are detailed in M. Iguernelala's PhD thesis [93].

3.3.4. Applications

- We have been quite successful in the application of Alt-Ergo to industrial development: qualification by Airbus France, integration of Alt-Ergo into the Spark Pro toolset.
- In the context of the BWare project, aiming at using Why3 and Alt-Ergo for discharging proof obligations generated by Atelier B, we made progress into several directions. The method of translation of B proof obligations into Why3 goals was first presented at ABZ'2012 [109]. Then, new drivers have been designed for Why3, in order to use new back-end provers Zenon modulo and iProver modulo. A notion of rewrite rule was introduced into Why3, and a transformation for simplifying goals before sending them to back-end provers was designed. Intermediate results obtained so far in the project were presented both at the French conference AFADL [81] and at ABZ'2014 [80].

  On the side of Alt-Ergo, recent developments have been made to efficiently discharge proof obligations generated by Atelier B. This includes a new plugin architecture to facilitate experiments with different SAT engines, new heuristics to handle quantified formulas, and important modifications in its internal data structures to boost performances of core decision procedures. Benchmarks realized on more than 10,000 proof obligations generated from industrial B projects show significant improvements [75].
- Hybrid automatons interleave continuous behaviors (described by differential equations) with discrete transitions. D. Ishii and G. Melquiond have worked on an automated procedure for verifying safety properties (that is, global invariants) of such systems [94].

3.3.5. Project-team Positioning

Automated Theorem Proving is a large community, but several sub-groups can be identified:

- The SMT-LIB community gathers people interested in reasoning modulo theories. In this community, only a minority of participants are interested in supporting first-order quantifiers at the same time as theories. SMT solvers that support quantifiers are Z3 (Microsoft Research Redmond, USA), CVC3 and its successor CVC4 [13].
- The TPTP community gathers people interested in first-order theorem proving.
- Other Inria teams develop provers: veriT by team Veridis, and Psyche by team Parsifal.
- Other groups develop provers dedicated to very specific cases, such as Metitarski [14] at Cambridge, UK, which aims at proving formulas on real numbers, in particular involving special functions such as log or exp. The goal is somewhat similar to our CoqInterval library, *cf* objective 4.

[13] http://cvc4.cs.nyu.edu/web/
[14] http://www.cl.cam.ac.uk/~lp15/papers/Arith/

It should be noticed that a large number of provers mentioned above are connected to Why3 as back-ends.

3.4. Formalization and Certification of Languages, Tools and Systems

Permanent researchers: S. Boldo, A. Charguéraud, C. Marché, G. Melquiond, C. Paulin

3.4.1. Real Numbers, Real Analysis, Probabilities

- S. Boldo, C. Lelay, and G. Melquiond have worked on the Coquelicot library, designed to be a user-friendly Coq library about real analysis [59], [60]. An easier way of writing formulas and theorem statements is achieved by relying on total functions in place of dependent types for limits, derivatives, integrals, power series, and so on. To help with the proof process, the library comes with a comprehensive set of theorems and some automation. We have exercised the library on several use cases: on an exam at university entry level [98], for the definitions and properties of Bessel functions [97], and for the solution of the one-dimensional wave equation [99]. We have also conducted a survey on the formalization of real arithmetic and real analysis in various proof systems [12].
- Watermarking techniques are used to help identify copies of publicly released information. They consist in applying a slight and secret modification to the data before its release, in a way that should remain recognizable even in (reasonably) modified copies of the data. Using the Coq ALEA library, which formalizes probability theory and probabilistic programs, D. Baelde together with P. Courtieu, D. Gross-Amblard from Rennes and C. Paulin have established new results about the robustness of watermarking schemes against arbitrary attackers [42]. The technique for proving robustness is adapted from methods commonly used for cryptographic protocols and our work illustrates the strengths and particularities of the ALEA style of reasoning about probabilistic programs.

3.4.2. Formalization of Languages, Semantics

- P. Herms, together with C. Marché and B. Monate (CEA List), has developed a certified VC generator, using Coq. The program for VC calculus and its specifications are both written in Coq, but the code is crafted so that it can be extracted automatically into a stand-alone executable. It is also designed in a way that allows the use of arbitrary first-order theorem provers to discharge the generated obligations [92]. On top of this generic VC generator, P. Herms developed a certified VC generator for C source code annotated using ACSL. This work is the main result of his PhD thesis [91].
- A. Tafat and C. Marché have developed a certified VC generator using Why3 [102], [103]. The challenge was to formalize the operational semantics of an imperative language, and a corresponding weakest precondition calculus, without the possibility to use Coq advanced features such as dependent types or higher-order functions. The classical issues with local bindings, names and substitutions were solved by identifying appropriate lemmas. It was shown that Why3 can offer a significantly higher amount of proof automation compared to Coq.
- A. Charguéraud, together with Alan Schmitt (Inria Rennes) and Thomas Wood (Imperial College), has developed an interactive debugger for JavaScript. The interface, accessible as a webpage in a browser, allows to execute a given JavaScript program, following step by step the formal specification of JavaScript developped in prior work on *JsCert* [50]. Concretely, the tool acts as a double-debugger: one can visualize both the state of the interpreted program and the state of the interpreter program. This tool is intended for the JavaScript committee, VM developpers, and other experts in JavaScript semantics.
- M. Clochard, C. Marché, and A. Paskevich have developed a general setting for developing programs involving binders, using Why3. This approach was successfully validated on two case studies: a verified implementation of untyped lambda-calculus and a verified tableaux-based theorem prover [69].

- M. Clochard, J.-C. Filliâtre, C. Marché, and A. Paskevich have developed a case study on the formalization of semantics of programming languages using Why3 [67]. This case study aims at illustrating recent improvements of Why3 regarding the support for higher-order logic features in the input logic of Why3, and how these are encoded into first-order logic, so that goals can be discharged by automated provers. This case study also illustrates how reasoning by induction can be done without need for interactive proofs, via the use of *lemma functions*.
- M. Clochard and L. Gondelman have developed a formalization of a simple compiler in Why3 [68]. It compiles a simple imperative language into assembler instructions for a stack machine. This case study was inspired by a similar example developed using Coq and interactive theorem proving. The aim is to improve significantly the degree of automation in the proofs. This is achieved by the formalization of a Hoare logic and a Weakest Precondition Calculus on assembly programs, so that the correctness of compilation is seen as a formal specification of the assembly instructions generated.

3.4.3. Project-team Positioning

The objective of formalizing languages and algorithms is very general, and it is pursued by several Inria teams. One common trait is the use of the Coq proof assistant for this purpose: Pi.r2 (development of Coq itself and its meta-theory), Gallium (semantics and compilers of programming languages), Marelle (formalization of mathematics), SpecFun (real arithmetic), Celtique (formalization of static analyzers).

Other environments for the formalization of languages include

- ACL2 system [15]: an environment for writing programs with formal specifications in first-order logic based on a Lisp engine. The proofs are conducted using a prover based on the Boyer-Moore approach. It is a rather old system but still actively maintained and powerful, developed at University of Texas at Austin. It has a strong industrial impact.
- Isabelle environment [16]: both a proof assistant and an environment for developing pure applicative programs. It is developed jointly at University of Cambridge, UK, Technische Universität München, Germany, and to some extent by the VALS team at LRI, Université Paris-Sud. It features highly automated tactics based on ATP systems (the Sledgehammer tool).
- The team "Trustworthy Systems" at NICTA in Australia [17] aims at developing highly trustable software applications. They developed a formally verified micro-kernel called seL4 [96], using a home-made layer to deal with C programs on top of the Isabelle prover.
- The PVS system [18] is an environment for both programming and proving (purely applicative) programs. It is developed at the Computer Science Laboratory of SRI international, California, USA. A major user of PVS is the team LFM [19] at NASA Langley, USA, for the certification of programs related to air traffic control.

In the Toccata team, we do not see these alternative environments as competitors, even though, for historical reasons, we are mainly using Coq. Indeed both Isabelle and PVS are available as back-ends of Why3.

3.5. Proof of Numerical Programs

Permanent researchers: S. Boldo, C. Marché, G. Melquiond

- Linked with objective 1 (Deductive Program Verification), the methodology for proving numerical C programs has been presented by S. Boldo in her habilitation [52] and as invited speaker [53]. An application is the formal verification of a numerical analysis program. S. Boldo, J.-C. Filliâtre, and G. Melquiond, with F. Clément and P. Weis (POMDAPI team, Inria Paris - Rocquencourt), and M. Mayero (LIPN), completed the formal proof of the second-order centered finite-difference scheme for the one-dimensional acoustic wave [55][4].

[15] http://www.cs.utexas.edu/~moore/acl2/
[16] http://isabelle.in.tum.de/
[17] http://ssrg.nicta.com.au/projects/TS/
[18] http://pvs.csl.sri.com/
[19] http://shemesh.larc.nasa.gov/fm/fm-main-team.html

- Several challenging floating-point algorithms have been studied and proved. This includes an algorithm by Kahan for computing the area of a triangle: S. Boldo proved an improvement of its error bound and new investigations in case of underflow [51]. This includes investigations about quaternions. They should be of norm 1, but due to the round-off errors, a drift of this norm is observed over time. C. Marché determined a bound on this drift and formally proved it correct [9]. P. Roux formally verified an algorithm for checking that a matrix is semi-definite positive [115]. The challenge here is that testing semi-definiteness involves algebraic number computations, yet it needs to be implemented using only approximate floating-point operations.
- Because of compiler optimizations (or bugs), the floating-point semantics of a program might change once compiled, thus invalidating any property proved on the source code. We have investigated two ways to circumvent this issue, depending on whether the compiler is a black box. When it is, T. Nguyen has proposed to analyze the assembly code it generates and to verify it is correct [112]. On the contrary, S. Boldo and G. Melquiond (in collaboration with J.-H. Jourdan and X. Leroy) have added support for floating-point arithmetic to the CompCert compiler and formally proved that none of the transformations the compiler applies modify the floating-point semantics of the program [58], [57].
- Linked with objectives 2 (Automated Reasoning) and 3 (Formalization and Certification of Languages, Tools and Systems), G. Melquiond has implemented an efficient Coq library for floating-point arithmetic and proved its correctness in terms of operations on real numbers [107]. It serves as a basis for an interval arithmetic on which Taylor models have been formalized. É. Martin-Dorel and G. Melquiond have integrated these models into CoqInterval [15]. This Coq library is dedicated to automatically proving the approximation properties that occur when formally verifying the implementation of mathematical libraries (libm).
- Double rounding occurs when the target precision of a floating-point computation is narrower than the working precision. In some situations, this phenomenon incurs a loss of accuracy. P. Roux has formally studied when it is innocuous for basic arithmetic operations [115]. É. Martin-Dorel and G. Melquiond (in collaboration with J.-M. Muller) have formally studied how it impacts algorithms used for error-free transformations [105]. These works were based on the Flocq formalization of floating-point arithmetic for Coq.
- By combining multi-precision arithmetic, interval arithmetic, and massively-parallel computations, G. Melquiond (in collaboration with G. Nowak and P. Zimmermann) has computed enough digits of the Masser-Gramain constant to invalidate a 30-year old conjecture about its closed form [108].

3.5.1. Project-team Positioning

This objective deals both with formal verification and floating-point arithmetic, which is quite uncommon. Therefore our competitors/peers are few. We may only cite the works by J. Duracz and M. Konečný, Aston University in Birmingham, UK.

The Inria team AriC (Grenoble - Rhône-Alpes) is closer to our research interests, but they are lacking manpower on the formal proof side; we have numerous collaborations with them. The Inria team Caramel (Nancy - Grand Est) also shares some research interests with us, though fewer; again, they do not work on the formal aspect of the verification; we have some occasional collaborations with them.

There are many formalization efforts from chip manufacturers, such as AMD (using the ACL2 proof assistant) and Intel (using the Forte proof assistants) but the algorithms they consider are quite different from the ones we study. The works on the topic of floating-point arithmetic from J. Harrison at Intel using HOL Light are really close to our research interests, but they seem to be discontinued.

A few deductive program verification teams are willing to extend their tools toward floating-point programs. This includes the KeY project and SPARK. We have an ongoing collaboration with the latter, in the context of the ProofInUSe project.

Deductive verification is not the only way to prove programs. Abstract interpretation is widely used, and several teams are interested in floating-point arithmetic. This includes the Inria team Antique (Paris - Rocquencourt) and a CEA List team, who have respectively developed the Astrée and Fluctuat tools. This approach targets a different class of numerical algorithms than the ones we are interested in.

Other people, especially from the SMT community (*cf* objective 2), are also interested in automatically proving formulas about floating-point numbers, notably at Oxford University. They are mainly focusing on pure floating-point arithmetic though and do not consider them as approximation of real numbers.

Finally, it can be noted that numerous teams are working on the verification of numerical programs, but assuming the computations are real rather than floating-point ones. This is out of the scope of this objective.

4. Application Domains

4.1. Safety-Critical Software

The application domains we target involve safety-critical software, that is where a high-level guarantee of soundness of functional execution of the software is wanted. Currently our industrial collaborations mainly belong to the domain of transportation, including aeronautics, railroad, space flight, automotive.

Verification of C programs, Alt-Ergo at Airbus Transportation is the domain considered in the context of the ANR U3CAT project, led by CEA, in partnership with Airbus France, Dassault Aviation, Sagem Défense et Sécurité. It included proof of C programs via Frama-C/Jessie/Why, proof of floating-point programs [104], the use of the Alt-Ergo prover via CAVEAT tool (CEA) or Frama-C/WP. Within this context, we contributed to a qualification process of Alt-Ergo with Airbus industry: the technical documents (functional specifications and benchmark suite) have been accepted by Airbus, and these documents were submitted by Airbus to the certification authorities (DO-178B standard) in 2012. This action is continued in the new project Soprano.

Certified compilation, certified static analyzers Aeronautics is the main target of the Verasco project, led by Verimag, on the development of certified static analyzers, in partnership with Airbus. This is a follow-up of the transfer of the CompCert certified compiler (Inria team Gallium) to which we contributed to the support of floating-point computations [58].

Transfer to the community of Ada development The former FUI project Hi-Lite, led by Adacore company, introduced the use of Why3 and Alt-Ergo as back-end to SPARK2014, an environment for verification of Ada programs. This is applied to the domain of aerospace (Thales, EADS Astrium). At the very beginning of that project, Alt-Ergo was added in the Spark Pro toolset (predecessor of SPARK2014), developed by Altran-Praxis: Alt-Ergo can be used by customers as an alternate prover for automatically proving verification conditions. Its usage is described in the new edition of the Spark book (Chapter "Advanced proof tools"). This action is continued in the new joint laboratory ProofInUse. A recent paper [65] provides an extensive list of applications of SPARK, a major one being the British air control management *iFacts*.

Transfer to the community of Atelier B In the current ANR project BWare, we investigate the use of Why3 and Alt-Ergo as an alternative back-end for checking proof obligations generated by *Atelier B*, whose main applications are railroad-related software [20], a collaboration with Mitsubishi Electric R&D Centre Europe (Rennes) (joint publication [109]) and ClearSy (Aix-en-Provence).

SMT-based Model-Checking: Cubicle S. Conchon (with A. Mebsout and F. Zaidi from VALS team at LRI) has a long-term collaboration with S. Krstic and A. Goel (Intel Strategic Cad Labs in Hillsboro, OR, USA) that aims in the development of the SMT-based model checker Cubicle (http://cubicle.lri.fr/) based on Alt-Ergo [106][7]. It is particularly targeted to the verification of concurrent programs and protocols.

[20] http://www.methode-b.com/en/links/

Apart from transportation, energy is naturally an application in particular with our long-term partner CEA, in the context of U3CAT and Soprano projects. We also indirectly target communications and data, in particular in contexts with a particular need for security or confidentiality: smart phones, Web applications, health records, electronic voting, etc. These are part of the applications of SPARK [65], including verification of security-related properties, including cryptographic algorithms. Also, our new AJACS project addresses issues related to security and privacy in web applications written in Javascript, also including correctness properties.

5. Highlights of the Year

5.1. Highlights of the Year

- S. Conchon: co-organizes POPL'2017 (January, Paris, http://conf.researchr.org/home/POPL-2017)
  - Major Int. Conference on Foundations of Programming Language, Semantics, Type Systems, Formal Proof Techniques

5.1.1. Awards

- [April 2016] Martin Clochard, Léon Gondelman, Mário Pereira: jointly receive the "Best student team" award of the *VerifyThis@ETAPS2016 verification competition*
- [July 2016] S. Boldo: Best Talk Award at workshop NSV *Computing a correct and tight rounding error bound using rounding-to-nearest*

6. New Software and Platforms

6.1. Alt-Ergo

Automated theorem prover for software verification
KEYWORDS: Software Verification - Automated theorem proving
FUNCTIONAL DESCRIPTION

Alt-Ergo is an automatic solver of formulas based on SMT technology. It is especially designed to prove mathematical formulas generated by program verification tools, such as Frama-C for C programs, or SPARK for Ada code. Initially developed in Toccata research team, Alt-Ergo's distribution and support are provided by OCamlPro since September 2013.

- Participants: Sylvain Conchon, Evelyne Contejean, Mohamed Iguernelala, Stephane Lescuyer and Alain Mebsout
- Partner: OCamlPro
- Contact: Sylvain Conchon
- URL: http://alt-ergo.lri.fr

6.2. CFML

Interactive program verification using characteristic formulae
KEYWORDS: Coq - Software Verification - Deductive program verification - Separation Logic
FUNCTIONAL DESCRIPTION

The CFML tool supports the verification of OCaml programs through interactive Coq proofs. CFML proofs establish the full functional correctness of the code with respect to a specification. They may also be used to formally establish bounds on the asymptotic complexity of the code. The tool is made of two parts: on the one hand, a characteristic formula generator implemented as an OCaml program that parses OCaml code and produces Coq formulae, and, on the other hand, a Coq library that provides notation and tactics for manipulating characteristic formulae interactively in Coq.

- Contact: Arthur Charguéraud
- URL: http://www.chargueraud.org/softs/cfml/

6.3. Coq

KEYWORDS: Proof - Certification - Formalisation
FUNCTIONAL DESCRIPTION

Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

- Participants: Benjamin Gregoire, Enrico Tassi, Bruno Barras, Yves Bertot, Pierre Boutillier, Xavier Clerc, Pierre Courtieu, Maxime Denes, Stephane Glondu, Vincent Gross, Hugo Herbelin, Pierre Letouzey, Assia Mahboubi, Julien Narboux, Jean-Marc Notin, Christine Paulin-Mohring, Pierre-Marie Pedrot, Loic Pottier, Matthias Puech, Yann Regis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack, Pierre-Yves Strub, Benjamin Werner, Guillaume Melquiond and Jean-Christophe Filliatre
- Partners: CNRS - ENS Lyon - Université Paris-Diderot - Université Paris-Sud
- Contact: Hugo Herbelin
- URL: http://coq.inria.fr/

6.4. CoqInterval

Interval package for Coq
KEYWORDS: Interval arithmetic - Coq
FUNCTIONAL DESCRIPTION

CoqInterval is a library for the proof assistant Coq. CoqInterval provides a method for proving automatically the inequality of two expression of real values.

The Interval package provides several tactics for helping a Coq user to prove theorems on enclosures of real-valued expressions. The proofs are performed by an interval kernel which relies on a computable formalization of floating-point arithmetic in Coq.

The Marelle team developed a formalization of rigorous polynomial approximation using Taylor models inside the Coq proof assistant, with a special focus on genericity and efficiency for the computations. In 2014, this library has been included in CoqInterval.

- Participants: Guillaume Melquiond, Erik Martin Dorel, Nicolas Brisebarre, Miora Maria Joldes, Micaela Mayero, Jean Michel Muller, Laurence Rideau and Laurent Thery
- Contact: Guillaume Melquiond
- URL: http://coq-interval.gforge.inria.fr/

6.5. Coquelicot

The Coquelicot library for real analysis in Coq
KEYWORDS: Coq - Real analysis

FUNCTIONAL DESCRIPTION

Coquelicot is library aimed for supporting real analysis in the Coq proof assistant. It is designed with three principles in mind. The first is the user-friendliness, achieved by implementing methods of automation, but also by avoiding dependent types in order to ease the stating and readability of theorems. This latter part was achieved by defining total function for basic operators, such as limits or integrals. The second principle is the comprehensiveness of the library. By experimenting on several applications, we ensured that the available theorems are enough to cover most cases. We also wanted to be able to extend our library towards more generic settings, such as complex analysis or Euclidean spaces. The third principle is for the Coquelicot library to be a conservative extension of the Coq standard library, so that it can be easily combined with existing developments based on the standard library.

- Participants: Sylvie Boldo, Catherine Lelay and Guillaume Melquiond
- Contact: Sylvie Boldo
- URL: http://coquelicot.saclay.inria.fr/

6.6. Cubicle

The Cubicle model checker modulo theories
KEYWORDS: Model Checking - Software Verification
FUNCTIONAL DESCRIPTION

Cubicle is an open source model checker for verifying safety properties of array-based systems, which corresponds to a syntactically restricted class of parametrized transition systems with states represented as arrays indexed by an arbitrary number of processes. Cache coherence protocols and mutual exclusion algorithms are typical examples of such systems.

- Participants: Sylvain Conchon and Alain Mebsout
- Contact: Sylvain Conchon
- URL: http://cubicle.lri.fr/

6.7. Flocq

The Flocq library for formalizing floating-point arithmetic in Coq
KEYWORDS: Floating-point - Arithmetic code - Coq
FUNCTIONAL DESCRIPTION

The Flocq library for the Coq proof assistant is a comprehensive formalization of floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties. It provides a framework for developers to formally certify numerical applications.

Flocq is currently used by the CompCert certified compiler for its support of floating-point computations.

- Participants: Guillaume Melquiond and Sylvie Boldo
- Contact: Sylvie Boldo
- URL: http://flocq.gforge.inria.fr/

6.8. Gappa

The Gappa tool for automated proofs of arithmetic properties
KEYWORDS: Floating-point - Arithmetic code - Software Verification - Constraint solving
FUNCTIONAL DESCRIPTION

Gappa is a tool intended to help verifying and formally proving properties on numerical programs dealing with floating-point or fixed-point arithmetic. It has been used to write robust floating-point filters for CGAL and it is used to certify elementary functions in CRlibm. While Gappa is intended to be used directly, it can also act as a backend prover for the Why3 software verification plateform or as an automatic tactic for the Coq proof assistant.

- Contact: Guillaume Melquiond
- URL: http://gappa.gforge.inria.fr/

6.9. Why3

The Why3 environment for deductive verification
KEYWORDS: Formal methods - Trusted software - Software Verification - Deductive program verification
FUNCTIONAL DESCRIPTION

Why3 is an environment for deductive program verification. It provides a rich language for specification and programming, called WhyML, and relies on external theorem provers, both automated and interactive, to discharge verification conditions. Why3 comes with a standard library of logical theories (integer and real arithmetic, Boolean operations, sets and maps, etc.) and basic programming data structures (arrays, queues, hash tables, etc.). A user can write WhyML programs directly and get correct-by-construction OCaml programs through an automated extraction mechanism. WhyML is also used as an intermediate language for the verification of C, Java, or Ada programs.

- Participants: Jean-Christophe Filliatre, Claude Marche, Guillaume Melquiond, Andriy Paskevych, François Bobot, Martin Clochard and Levs Gondelmans
- Partners: CNRS - Université Paris-Sud
- Contact: Claude Marche
- URL: http://why3.lri.fr/

7. New Results

7.1. Deductive Verification

A bit-vector library for deductive verification. C. Fumex and C. Marché developed a new library for bit-vectors, in Why3 and SPARK. This library is rich enough for the formal specification of functional behavior of programs that operate at the level of bits. It is also designed to exploit efficiently the support for bit-vectors built-in in some SMT solvers. This work is done in the context of the ProofInUse joint laboratory. The SPARK front-end of Why3, for the verification of Ada programs, is extended to exploit this new bit-vector theory. Several cases studies are conducted: efficient search for rightmost bit of a bit-vector, efficient computation of the number of bits set to 1, efficient solving of the n-queens problem. At the level of SPARK, a program inspired from some industrial code (originally developed in C par J. Gerlach, Fraunhofer FOKUS Institute, Germany and partially proved with Frama-C and Coq) is specified in SPARK and proved with automatic solvers only. A paper on that library together with the way it is connected with the built-in support for bitvectors in SMT solver was presented at the NASA Formal methods Conference [24]. The support for bit-vectors is distributed with SPARK since 2015, and SPARK users already reported that several verification conditions, that couldn't be proved earlier, are now proved automatically.

Counterexamples from proof failures. D. Hauzar and C. Marché worked on counterexample generation from failed proof attempts. They designed a new approach for generating potential counterexamples in the deductive verification setting, and implemented in Why3. When the logic goal generated for a given verification condition is not shown unsatisfiable by an SMT solvers, some solver can propose a model. By carefully reverting the transformation chain (from an input program through the VC generator and the various translation steps to solvers), this model is turned into a potential counterexample that the user can exploit to analyze why its original code is not proved. The approach is implemented in the chain from Ada programs through SPARK, Why3, and SMT solvers CVC4 and Z3. This work is described in a research report [35] and a paper was rpesented at the SEFM Conference [25]. The work on the implementation was continued by S. Dailler. It was considered robust enough to be distributed in the release Pro 16 of SPARK.

Static versus dynamic verification. C. Marché, together with Y. Moy from AdaCore, J. Signoles and N. Kosmatov from CEA-LIST, wrote a survey paper about the design of the specification languages of Why3 and its front-ends Frama-C and SPARK. The choices made when designing these specification languages differ significantly, in particular with respect to the executability of specifications. The paper reviews these differences and the issues that result from these choices. The paper also emphasizes two aspects where static and dynamic aspects of the specifications play an important role: the specific feature of *ghost code*, and the techniques that help users understand why static verification fails. This paper was presented at the Isola Symposium [26].

Higher-Order Representation Predicates. A. Charguéraud investigated how to formalize in Separation Logic representation predicates for describing mutable container data structures that store mutable elements that are themselves described using representation predicates. (In Separation Logic, representation predicates are used to describe mutable data structures, by establishing a relationship between the entry point of the structure, the piece of heap over which this structure spans, and the logical model associated with the structure.) The solution proposed, based on "higher-order representation predicates", allows for concise specifications of such containers. A. Charguéraud has published a paper presenting, through a collection of practical examples, solutions to the challenges associated with verification proofs based on higher-order representation predicates [19].

Temporary Read-Only Permissions for Separation Logic A. Charguéraud and François Pottier (Inria Paris) have developed an extension of Separation Logic with temporary read-only permissions. This mechanism allows to temporarily convert any assertion (or "permission") to a read-only form. Unlike with fractional permissions, no accounting is required: the proposed read-only permissions can be freely duplicated and discarded. Where mutable data structures are temporarily accessed only for reading, the proposed read-only permissions enable more concise specifications and proofs. All the metatheory is verified in Coq. An article has been submitted to a conference [20].

Reasoning About Iteration. J.-C. Filliâtre and M. Pereira proposed a new approach to the problem of specifying iteration, verifying iterators (such as cursors or higher-order functions), and using iterators. The idea is to characterize the sequence of elements enumerated so far, and only those. The proof methodology is modular, iterator implementations and clients being verified independently of each other. The proposed method is validated experimentally in Why3. This work has been published first at JFLA 2016 [33] and then at NFM 2016 [22]. A journal version of this work is under submission.

Defunctionalization for proving higher-order programs. J.-C. Filliâtre and M. Pereira proposed a new approach to the verification of higher-order programs, using the technique of defunctionalization, that is, the translation of first-class functions into first-order values. This is an early experimental work, conducted on examples only within the Why3 system. This work has been published at JFLA 2017 [30].

A Type System for Deductive Verification. J.-C. Filliâtre, L. Gondelman, and A. Paskevich proposed a practical method to track pointer aliases statically in a large family of computer programs. Their approach relies on a special type system with singleton regions and effects which both can be

inferred automatically, without requiring additional user annotations. This kind of static analysis is important for deductive program verification, since it allows us to construct verification conditions using the traditional rules in the spirit of Hoare and Dijkstra, without recurring to more sophisticated solutions (memory models, separation logic) which incur additional complexity both for a user and a verification tool. The proposed method is implemented in Why3 and described in a technical report [37].

Ghost Code. J.-C. Filliâtre, L. Gondelman, and A. Paskevich published a paper on a general approach to the concept of ghost code in the journal of *Formal Methods in System Design* [14]. Ghost code is a subset of program code that serves the purposes of specification and verification: it can be erased from the program without affecting its result. This work forms the basis of the support for ghost code in Why3. This work is an extended version of the paper presented at the 26th International Conference on Computer Aided Verification (CAV) in 2014.

7.2. Automated Reasoning

Decision Procedures via Axiomatizations with Triggers. C. Dross, A. Paskevich, J. Kanig and S. Conchon published a paper in the *Journal of Automated Reasoning* [13] about integration of first-order axiomatizations with triggers as decision procedures in an SMT solver. This work extends a part of C. Dross PhD thesis [83]. A formal semantics of the notion of trigger is presented, with a general setting to show how a first-order axiomatization with triggers can be proved correct, complete, and terminating. An extended DPLL(T) algorithm can then integrate such an axiomatization with triggers, as a decision procedure for the theory it defines.

Lightweight Approach for Declarative Proofs. M. Clochard designed an extension of first-order logic, for describing reasoning steps needed to discharge a proof obligation. The extension is under the form of two new connectives, called proof indications, that allow the user to encode reasoning steps inside a logic formula. This extension makes possible to use the syntax of formulas as a proof language. The approach was presented at the JFLA conference [29] and implemented in Why3. It brings a lightweight mechanism for declarative proofs in an environment like Why3 where provers are used as black boxes. Moreover, this mechanism restricts the scope of auxiliary lemmas, reducing the size of proof obligations sent to external provers.

7.3. Certification of Algorithms, Languages, Tools and Systems

Case study: Matrix Multiplication. M. Clochard, L. Gondelman and M. Pereira wrote a paper describing a complete solution for the first challenge of the VerifyThis 2016 competition held at the 18th ETAPS Forum, where they obtain the award for the best student team. Two variants for the multiplication of matrices are presented and proved: a naive version using three nested loops and Strassen's algorithm. To formally specify the two multiplication algorithms, they developed a new Why3 theory of matrices, and they applied a reflection methodology to condut some of the proofs. This work was presented at the VSTTE Conference [21]. An extended version that considers arbitrary rectangular martices instead of square ones is in preparation. The development is available at http://toccata.lri.fr/gallery/verifythis_2016_matrix_multiplication.en.html.

Case study: Koda-Ruskey's algorithm for generating ideals of a forest. J.-C. Filliâtre and M. Pereira presented the first formal proof of an implementation of Koda and Ruskey's algorithm (an algorithm for generating all ideals of a forest poset as a Gray code) at VSTTE 2016 [23]. The proof is conducted within the Why3 system and is mostly automatic.

The Lax-Milgram Theorem. S. Boldo, F. Clément, F. Faissole, V. Martin, and M. Mayero have worked on a Coq formal proof of the Lax–Milgram theorem. The Finite Element Method is a widely-used method to solve numerical problems coming for instance from physics or biology. To obtain the highest confidence on the correction of numerical simulation programs implementing the Finite Element Method, one has to formalize the mathematical notions and results that allow to establish the sound-ness of the method. The Lax–Milgram theorem may be seen as one of those

theoretical cornerstones: under some completeness and coercivity assumptions, it states existence and uniqueness of the solution to the weak formulation of some boundary value problems. This article presents the full formal proof of the Lax–Milgram theorem in Coq. It requires many results from linear algebra, geometry, functional analysis , and Hilbert spaces. This has been published at the 6th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2017) [18].

ALEA library extended with continuous datatypes The ALEA library uses a monadic construction to formalize discrete measure theory. F. Faissole and B. Spitters proposed to extend it to continuous datatypes. They used both synthetic topology and homotopy type theory to achieve the formalization. This work is presented at the Workshop on Coq for Programming Languages [32].

Case study: Strongly Connected Components of a Graph R. Chen and J.-J. Lévy designed a formal proof of Tarjan's algorithm for computing the strongly connected component of a directed graph. The proof is conducted using Why3. This work is presented at the JFLA conference [28]. This case study is part of a larger set of case studies on algorithms on graphs http://pauillac.inria.fr/~levy/why3/.

Case study: Unix Pathname Resolution R. Chen, M. Clochard and C.-Marché designed a formal proof of an algorithm for resolving a pathname in Unix file systems. The proof is conducted using Why3 [34]. This case study is part of the CoLiS project.

7.4. Floating-Point and Numerical Programs

-

Interval arithmetic and Taylor models. É. Martin-Dorel and G. Melquiond have worked on integrating the CoqInterval and CoqApprox libraries into a single package. The CoqApprox library is dedicated to computing verified Taylor models of univariate functions so as to compute approximation errors. The CoqInterval library reuses this work to automatically prove bounds on real-valued expressions. A large formalization effort took place during this work, so as to get rid of all the holes remaining in the formal proofs of CoqInterval. It was also the chance to perform a comparison between numerous decision procedures dedicated to proving nonlinear inequalities involving elementary functions. This work has been published in the *Journal of Automated Reasoning* [15].

Interval arithmetic and univariate integrals. A. Mahboubi, G. Melquiond, and T. Sibut-Pinote have extended the CoqInterval library with support for definite univariate integrals. The library is now able to automatically and formally verify bounds on the value of integrals by computing rigorous polynomial approximations of integrands. This work has been presented at the 7th International Conference on Interactive Theorem Proving [27].

Robustness of 2Sum and Fast2Sum. S. Boldo, S.Graillat, and J.-M. Muller have worked on the 2Sum and Fast2Sum algorithms, that are important building blocks in numerical computing. They are used (implicitely or explicitely) in many compensated algorithms or for manipulating floating-point expansions. They showed that these algorithms are much more robust than it is usually believed: the returned result makes sense even when the rounding function is not round-to-nearest, and they are almost immune to overflow. This work has been submitted [36].

Computing error bounds without changing the rouding mode. S. Boldo has created an algorithm to compute a correct and tight rounding error bound for a floating-point computation. The rounding error can be bounded by folklore formulas, such as $\varepsilon|x|$ or $\varepsilon|\circ(x)|$. This gets more complicated when underflow is taken into account. To compute this error bound in practice, a directed rounding is usually used. This work describes an algorithm that computes a correct bound using only rounding to nearest, therefore without requiring a costly change of the rounding mode. This is formally proved using the Coq formal proof assistant to increase the trust in this algorithm. This has been published at the 9th International Workshop on Numerical Software Verification [17].

Floating-Point Computations and Iterators. S. Boldo has worked on the formal verification of a floating-point case study where the common iterators `fold_left` and `fold_right` have not the wanted behaviors. She then had to define other iterators, which are very similar in most cases, but that do behave well in our case study. This has been published at the 1st Workshop on High-Consequence Control Verification [31].

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. ProofInUse Joint Laboratory

Participants: Claude Marché [contact], Jean-Christophe Filliâtre, Andrei Paskevich.

ProofInUse is a joint project between the Toccata team and the SME AdaCore. It was selected and funded by the ANR programme "Laboratoires communs", starting from April 2014, for 3 years http://www.spark-2014.org/proofinuse.

The SME AdaCore is a software publisher specializing in providing software development tools for critical systems. A previous successful collaboration between Toccata and AdaCore enabled *Why3* technology to be put into the heart of the AdaCore-developed SPARK technology.

The goal is now to promote and transfer the use of deduction-based verification tools to industry users, who develop critical software using the programming language Ada. The proof tools are aimed at replacing or complementing the existing test activities, whilst reducing costs.

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. ELFIC

Participants: Sylvie Boldo [contact], Claude Marché, Guillaume Melquiond.

ELFIC is a working group of the Digicosme Labex. S. Boldo is the principal investigator. It began in 2014 for one year and was extended for one year. https://digicosme.lri.fr/GT+ELFIC

The ELFIC project focuses on proving the correctness of the FELiScE (Finite Elements for Life Sciences and Engineering) C++ library which implements the finite element method for approximating solutions to partial differential equations. Finite elements are at the core of numerous simulation programs used in industry. The formal verification of this library will greatly increase confidence in all the programs that rely on it. Verification methods developed in this project will be a breakthrough for the finite element method, but more generally for the reliability of critical software relying on intricate numerical algorithms.

Partners: Inria team Pomdapi; Ecole Polytechnique, LIX; CEA LIST; Université Paris 13, LIPN; UTC, LMAC (Compiègne).

9.1.2. ELEFFAN

Participant: Sylvie Boldo [contact].

ELEFFAN is a Digicosme project funding the PhD of F. Faissole. S. Boldo is the principal investigator. It began in 2016 for three years. https://project.inria.fr/eleffan/

The ELEFFAN project aims at formally proving rounding error bounds of numerical schemes.

Partners: ENSTA Paristech (A. Chapoutot)

9.2. National Initiatives

9.2.1. ANR CoLiS

Participants: Claude Marché [contact], Andrei Paskevich.

The CoLiS research project is funded by the programme "Société de l'information et de la communication" of the ANR, for a period of 48 months, starting on October 1st, 2015. http://colis.irif.univ-paris-diderot.fr/

The project aims at developing formal analysis and verification techniques and tools for scripts. These scripts are written in the POSIX or bash shell language. Our objective is to produce, at the end of the project, formal methods and tools allowing to analyze, test, and validate scripts. For this, the project will develop techniques and tools based on deductive verification and tree transducers stemming from the domain of XML documents.

Partners: Université Paris-Diderot, IRIF laboratory (formerly PPS & LIAFA), coordinator ; Inria Lille, team LINKS

9.2.2. ANR Vocal

Participants: Jean-Christophe Filliâtre [contact], Andrei Paskevich.

The Vocal research project is funded by the programme "Société de l'information et de la communication" of the ANR, for a period of 48 months, starting on October 1st, 2015. https://vocal.lri.fr/

The goal of the Vocal project is to develop the first formally verified library of efficient general-purpose data structures and algorithms. It targets the OCaml programming language, which allows for fairly efficient code and offers a simple programming model that eases reasoning about programs. The library will be readily available to implementers of safety-critical OCaml programs, such as Coq, Astrée, or Frama-C. It will provide the essential building blocks needed to significantly decrease the cost of developing safe software. The project intends to combine the strengths of three verification tools, namely Coq, Why3, and CFML. It will use Coq to obtain a common mathematical foundation for program specifications, as well as to verify purely functional components. It will use Why3 to verify a broad range of imperative programs with a high degree of proof automation. Finally, it will use CFML for formal reasoning about effectful higher-order functions and data structures making use of pointers and sharing.

Partners: team Gallium (Inria Paris-Rocquencourt), team DCS (Verimag), TrustInSoft, and OCamlPro.

9.2.3. ANR Ajacs

Participant: Arthur Charguéraud [contact].

The AJACS research project is funded by the programme "Société de l'information et de la communication" of the ANR, for a period of 42 months, starting on October 1st, 2014. http://ajacs.inria.fr/

The goal of the AJACS project is to provide strong security and privacy guarantees on the client side for web application scripts implemented in JavaScript, the most widely used language for the Web. The proposal is to 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 deriving more precise analyses. Another aspect of the proposal is the design and certification of security and privacy enforcement mechanisms for web applications, including the APIs used to program real-world applications. On the Toccata side, the focus will be on the formalization of secure subsets of JavaScript, and on the mechanization of proofs of translations from high-level languages into JavaScript.

Partners: team Celtique (Inria Rennes - Bretagne Atlantique), team Prosecco (Inria Paris - Rocquencourt), team Indes (Inria Sophia Antipolis - Méditerranée), and Imperial College (London).

9.2.4. ANR FastRelax

Participants: Sylvie Boldo [contact], Guillaume Melquiond.

This is a research project funded by the programme "Ingénierie Numérique & Sécurité" of the ANR. It is funded for a period of 48 months and it has started on October 1st, 2014. http://fastrelax.gforge.inria.fr/

Our aim is to develop computer-aided proofs of numerical values, with certified and reasonably tight error bounds, without sacrificing efficiency. Applications to zero-finding, numerical quadrature or global optimization can all benefit from using our results as building blocks. We expect our work to initiate a "fast and reliable" trend in the symbolic-numeric community. This will be achieved by developing interactions between our fields, designing and implementing prototype libraries and applying our results to concrete problems originating in optimal control theory.

Partners: team ARIC (Inria Grenoble Rhône-Alpes), team MARELLE (Inria Sophia Antipolis - Méditerranée), team SPECFUN (Inria Saclay - Île-de-France), Université Paris 6, and LAAS (Toulouse).

9.2.5. ANR Soprano

Participants: Sylvain Conchon [contact], Guillaume Melquiond.

The Soprano research project is funded by the programme "Sciences et technologies logicielles" of the ANR, for a period of 42 months, starting on October 1st, 2014. http://soprano-project.fr/

The SOPRANO project aims at preparing the next generation of verification-oriented solvers by gathering experts from academia and industry. We will design a new framework for the cooperation of solvers, focused on model generation and borrowing principles from SMT (current standard) and CP (well-known in optimization). Our main scientific and technical objectives are the following. The first objective is to design a new collaboration framework for solvers, centered around synthesis rather than satisfiability and allowing cooperation beyond that of Nelson-Oppen while still providing minimal interfaces with theoretical guarantees. The second objective is to design new decision procedures for industry-relevant and hard-to-solve theories. The third objective is to implement these results in a new open-source platform. The fourth objective is to ensure industrial-adequacy of the techniques and tools developed through periodical evaluations from the industrial partners.

Partners: team DIVERSE (Inria Rennes - Bretagne Atlantique), Adacore, CEA List, Université Paris-Sud, and OCamlPro.

9.2.6. ANR CAFEIN

Participant: Sylvain Conchon [contact].

The CAFEIN research project is funded by the programme "Ingénierie Numérique & Sécurité" of the ANR, for a period of 3 years, starting on February 1st, 2013. https://cavale.enseeiht.fr/CAFEIN/.

This project addresses the formal verification of functional properties at specification level, for safety critical reactive systems. In particular, we focus on command and control systems interacting with a physical environment, specified using the synchronous language Lustre.

A first goal of the project is to improve the level of automation of formal verification, by adapting and combining existing verification techniques such as SMT-based temporal induction, and abstract interpretation for invariant discovery. A second goal is to study how knowledge of the mathematical theory of hybrid command and control systems can help the analysis at the controller's specification level. Third, the project addresses the issue of implementing real valued specifications in Lustre using floating-point arithmetic.

Partners: ONERA, CEA List, ENSTA, teams Maxplus (Inria Saclay - Île-de-France), team Parkas (Inria Paris - Rocquencourt), Perpignan University, Prover Technology, Rockwell Collins.

9.2.7. ANR BWare

Participants: Sylvain Conchon [contact], Jean-Christophe Filliâtre, Andrei Paskevich, Claude Marché.

The BWare research project is funded by the programme "Ingénierie Numérique & Sécurité" of the ANR, a period of 4 years, starting on September 1st, 2012. http://bware.lri.fr.

BWare is an industrial research project that aims to provide a mechanized framework to support the automated verification of proof obligations coming from the development of industrial applications using the B method and requiring high guarantee of confidence. The methodology used in this project consists of building a generic platform of verification relying on different theorem provers, such as first-order provers and SMT solvers. The variety of these theorem provers aims at allowing a wide panel of proof obligations to be automatically verified by the platform. The major part of the verification tools used in BWare have already been involved in some experiments, which have consisted in verifying proof obligations or proof rules coming from industrial applications [109]. This therefore should be a driving factor to reduce the risks of the project, which can then focus on the design of several extensions of the verification tools to deal with a larger amount of proof obligations.

The partners are: Cedric laboratory at CNAM (CPR Team, project leader); teams Gallium and Deducteam (Inria Paris - Rocquencourt) ; Mitsubishi Electric R&D Centre Europe, ClearSy (the company which develops and maintains *Atelier B*), and the start-up OCamlPro.

9.2.8. ANR Verasco

Participants: Guillaume Melquiond [contact], Sylvie Boldo, Arthur Charguéraud, Claude Marché.

The Verasco research project is funded by the programme "Ingénierie Numérique & Sécurité" of the ANR, for a period of 4 years and a half, starting on January 1st, 2012. Project website: http://verasco.imag.fr.

The main goal of the project is to investigate 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. More precisely, the project aims at developing a generic static analyzer based on abstract interpretation for the C language, along with a number of advanced abstract domains and domain combination operators, and prove the soundness of this analyzer using the *Coq* proof assistant. Likewise, the project keeps working on the CompCert C formally-verified compiler, the first realistic C compiler that has been mechanically proved to be free of miscompilation, and carry it to the point where it could be used in the critical software industry.

Partners: teams Gallium and Abstraction (Inria Paris - Rocquencourt), Airbus avionics and simulation (Toulouse), IRISA (Rennes), Verimag (Grenoble).

9.2.9. FUI LCHIP

Participant: Sylvain Conchon [contact].

LCHIP (Low Cost High Integrity Platform) is aimed at easing the development of safety critical applications (up to SIL4) by providing: (i) a complete IDE able to automatically generate and prove bounded complexity software (ii) a low cost, safe execution platform. The full support of DSLs and third party code generators will enable a seamless deployment into existing development cycles. LCHIP gathers scientific results obtained during the last 20 years in formal methods, proof, refinement, code generation, etc. as well as a unique return of experience on safety critical systems design. http://www.clearsy.com/en/2016/10/4260/

Partners: 2 technology providers (ClearSy, OcamlPro), in charge of building the architecture of the platform ; 3 labs (IFSTTAR, LIP6, LRI), to improve LCHIP IDE features ; 2 large companies (SNCF, RATP), representing public ordering parties, to check compliance with standard and industrial railway use-case.

The project lead by ClearSy has started in April 2016 and lasts 3 years. It is funded by BpiFrance as well as French regions.

9.2.10. ANR PARDI

Participant: Sylvain Conchon [contact].

Verification of parameterized distributed systems, 2016-2021.

Partners: Université Paris VI - Université Paris XI - Inria NANCY

9.3. European Initiatives

9.3.1. FP7 & H2020 Projects

Project acronym: ERC Deepsea

Project title: Parallel dynamic computations

Duration: Jun. 2013 - Jun. 2018

Coordinator: Umut A. Acar

Other partners: Carnegie Mellon University

Abstract:

The objective of this project is to develop abstractions, algorithms and languages for parallelism and dynamic parallelism with applications to problems on large data sets. Umut A. Acar (affiliated to Carnegie Mellon University and Inria Paris - Rocquencourt) is the principal investigator of this ERC-funded project. The other main researchers involved are Mike Rainey (Inria, Gallium team), who is full-time on the project, and Arthur Charguéraud (Inria, Toccata team), who works 40% of his time to the project. Project website: http://deepsea.inria.fr/.

9.3.2. Collaborations in European Programs, Except FP7 & H2020

Program: COST (European Cooperation in Science and Technology).

Project acronym: EUTypes https://eutypes.cs.ru.nl/

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

Duration: 2015-2019

Coordinator: Herman Geuvers, Radboud University Nijmegen, The Netherlands

Other partners: 36 members countries, see http://www.cost.eu/COST_Actions/ca/CA15123?parties

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

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

9.3.3. Collaborations with Major European Organizations

Imperial College London (UK)

Certification of JavaScript, AJACS project

9.4. International Research Visitors

9.4.1. Visits of International Scientists

- Ran Chen is a PhD student from Institute of Software (Chinese Academy of Sciences, Beijing, China) visiting the team for 10 months under the supervision of C. Marché and J.-J. Lévy (PiR2 team, Inria Paris). She is working on the formal verification of graphs algorithms, and also in the context of the CoLiS project on verification of some aspects of the Unix file system and shell scripts. [34]
- Cláudio Belo Lourenço is a PhD student from Universidade do Minho, Portugal. He studies deductive verification of imperative programs and the behaviour of different kinds of verification condition generators [101]. The goal of his visit is to use Why3 as a platform for prototyping and experimental evaluation of these generators.

9.4.2. Visits to International Teams

9.4.2.1. Research Stays Abroad

- F. Faissole has spent two months visiting B. Spitters at Aarhus University (Denmark) They proposed an extension of ALEA library to continuous datatypes.[32].
- M. Roux has spent three months with D. Jovanovic and B. Dutertre at SRI (California, USA). They worked on extending Sally, the new model checker of SRI based on SAL, to add the verification of parameterized cache coherence protocols. The software can be found on https://github.com/SRI-CSL/sally.
- S. Conchon has been invited a month at SRI by D. Jovanovic. During this visit, he has collaborated with CSL researchers to compare the design and implementation choices between the model checkers Sally and Cubicle.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Organisation

10.1.1.1. General Chair, Scientific Chair

- S. Boldo, vice-president of the 28th "Journées Francophones des Langages Applicatifs" (JFLA 2017)
- J.-C. Filliâtre, organizer of EJCP (École Jeunes Chercheurs en Programmation du GDR GPL) at Lille on June 27–July 1, 2016. 42 participants. http://ejcp2016.univ-lille1.fr/
- A. Paskevich, program chair of the 9th Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2017), in collaboration with Thomas Wies (NYU).

10.1.1.2. Member of the Organizing Committees

- S. Conchon, local chair for the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2017), held in Paris, France in January 2017.
- G. Melquiond, web chair for the 23rd IEEE Symposium on Computer Arithmetic (Arith 23), held in Silicon Valley, USA in July 2016.
- S. Boldo, member of the organization committee of the Inria Scientific Days in Rennes (June 2016).

10.1.2. Scientific Events Selection

10.1.2.1. Member of the Conference Program Committees

- S. Boldo, PC of the 23rd IEEE Symposium on Computer Arithmetic (ARITH 2016).
- S. Boldo, PC of the 9th International Workshop on Numerical Software Verification (NSV 2016).
- S. Boldo, PC of the 1st Workshop on High-Consequence Control Verification (HCCV 2016).
- S. Boldo, PC of the 27th "Journées Francophones des Langages Applicatifs" (JFLA 2016).
- A. Charguéraud, PC of the International Workshop on Hammers for Type Theories (HaTT 2016).
- A. Charguéraud, PC of the Workshop on ML (ML 2016).
- G. Melquiond, PC of the 3rd International Workshop on Coq for Programming Languages (CoqPL 2017).

10.1.2.2. Reviewer

The members of the Toccata team have reviewed papers for numerous international conferences.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

- G. Melquiond is a member of the editorial board of *Reliable Computing*.
- S. Boldo is member of the editorial board of Binaire http://binaire.blog.lemonde.fr, the blog of the French Computer Science Society.
- J.-C. Filliâtre is member of the editorial board of the *Journal of Functional Programming*.
- C. Paulin is member of the editorial board of the *Journal of Formalized Reasoning*.

10.1.3.2. Reviewer - Reviewing Activities

The members of the Toccata team have reviewed numerous papers for numerous international journals.

10.1.4. Invited Talks

- S. Boldo gave an invited lecture at Effective Analysis: Foundations, Implementations, Certification in January 2016 at Marseille, France.
- S. Boldo gave a keynote talk in Cambridge at a local workshop about Testing and Verification in Computational Science.
- A. Charguéraud gave an invited talk at the Royal Society specialist meeting "Verified trustworthy software systems", presenting an interactive interpreter for the semantics of JavaScript, in London, on April 7th.
- S. Conchon gave an invited lecture "Model Checking Modulo Theories with Cubicle" at the 2016 edition of the School for Junior Researchers in Programming (École Jeunes Chercheurs en Programmation, EJCP 2016, http://ejcp2016.univ-lille1.fr/) held in Lille, France.
- A. Paskevich gave an invited lecture "Deductive Program Verification using Why3" at the 2016 edition of the School for Junior Researchers in Programming (École Jeunes Chercheurs en Programmation, EJCP 2016, http://ejcp2016.univ-lille1.fr/) held in Lille, France.

10.1.5. Leadership within the Scientific Community

- C. Paulin, scientific leader of Labex DigiCosme http://labex-digicosme.fr (Digital Worlds: distributed data, programs and architectures), until June 2016. It is a project launched by the French Ministry of research and higher education as part of the program "Investissements d'avenir", it involves the 14 research units in computer science and communications from the "Paris-Saclay" cluster.
- C. Paulin, dean of the Faculty of Sciences of Université Paris-Sud, since July 2016.

10.1.6. Scientific Expertise

- S. Boldo, member of the reviewing board for the ANR (first step in 2016).
- S. Boldo, member of the 2016 committee for the Gilles Kahn PhD award of the French Computer Science Society.
- S. Conchon and A. Paskevich, members of the *"commission consultative de spécialistes de l'université"*, Section 27, University Paris-Sud since December 2014.
- C. Marché, president of the evaluation committee of the joint Digiteo-DigiCosme call for projects https://digicosme.lri.fr/AAPDigiteoDigiCosme2016. The committee selected 10 thesis projects for funding, among 52 submissions. The committee also selected to support 10 scientific events in the Île-de-France region.
- C. Marché, member of the scientific commission of Inria-Saclay, in charge of selecting candidates for PhD grants, Post-doc grants, temporary leaves from universities ("délégations")
- C. Marché, member of the "Bureau du Comité des Projets" of Inria-Saclay, in charge of examining proposals for creation of new Inria project-teams.

- C. Marché, member of a hiring committee for an associate professor position in computer science at CNAM, Paris, France. (sep-oct 2016)

10.1.7. Research Administration

- S. Boldo, member of the CCD, *commission consultative des doctorants*.
- S. Boldo, member of the CLFP, *comité local de formation permanente*.
- S. Boldo, scientific head for Saclay for the MECSI group for networking about computer science popularization inside Inria.
- A. Charguéraud is vice-president of *France-ioi*, a non-profit organization in charge of the selection and the training of the French team to the International Olympiads in Informatics (IOI). France-ioi also provides online exercises in programming and algorithmics—in average, over 100,000 such exercises are solved every month on the website.
- A. Charguéraud is a board member of the non-profit organization *Animath*, which aims at developing interest in mathematics among young students.
- A. Charguéraud and G. Melquiond are members of the committee for the monitoring of PhD students (*"commission de suivi des doctorants"*).
- J.-C. Filliâtre is *correcteur au concours d'entrée à l'École Polytechnique et aux ENS* (computer science examiner for the entrance exam at École Polytechnique and Écoles Normales Supérieures) since 2008.
- C. Marché, director of the ProofInUse Joint Laboratory between Inria and AdaCore, http://www.spark-2014.org/proofinuse
- C. Paulin, member of the *"commission consultative de spécialistes de l'université"*, Section 27, University Paris-Sud since April 2010. C. Paulin is the president of this committee since December 2014.
- C. Paulin, chaired the hiring committee for a professor position in computer science at Université Paris-Sud.
- J.-C. Filliâtre, member of the board of GDR GPL, since January 2016.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Master Parisien de Recherche en Informatique (MPRI) https://wikimpri.dptinfo.ens-cachan.fr/doku.php: "Proofs of Programs" http://www.lri.fr/~marche/MPRI-2-36-1/ (M2), C. Marché (12h), A. Charguéraud (12h), Université Paris-Diderot, France.

Master: Fondements de l'informatique et ingénierie du logiciel (FIIL) https://www.lri.fr/~conchon/parcours_fiil/: "Software Model Checking" (M2), S. Conchon (9h), "Programmation C++11 avancée" (M2), G. Melquiond (12h), "Vérification déductive de programmes" (M2), A. Paskevich (10.5h), Université Paris-Sud, France.

DUT (Diplôme Universitaire de Technologie): M1101 "Introduction aux systèmes informatiques", A. Paskevich (36h), M3101 "Principes des systèmes d'exploitation", A. Paskevich (58.5h), IUT d'Orsay, Université Paris-Sud, France.

Licence: "Langages de programmation et compilation" (L3), J.-C. Filliâtre (26h), École Normale Supérieure, France.

Licence: "INF411: Les bases de l'algorithmique et de la programmation" (L3), J.-C. Filliâtre (16h), École Polytechnique, France.

Licence: "Programmation fonctionnelle avancée" (L3), S. Conchon (45h), Université Paris-Sud, France.

Licence: “Introduction à la programmation fonctionnelle” (L2), S. Conchon (25h), Université Paris-Sud, France.

10.2.2. Internships

- Raphaël Rieu-Helft (ENS, Paris) is a pre-PhD student doing an internship for 6 months under supervision of C. Marché, G. Melquiond and A. Paskevich. He is working on the design and the formal verification of a library for unbounded integer arithmetic. Why3 is used for formally verifying the functional behaviour of the library operations. Raphaël is also implementing in Why3 a mechanism for extracting code to the C language, in order to obtain a certified code that runs very efficiently.
- Lucas Baudin (ENS, Paris) is a master 1 intern under the supervision of J.-C. Filliâtre between September 2016 and January 2017. He is working on the inference of loop invariants by abstract interpretation in the tool Why3.
- F. Faissole was a master 2 trainee under the supervision of S. Boldo between March and August 2016. He worked on the formal proof of the Lax-Milgram theorem.

10.2.3. Supervision

PhD: L. Gondelmans, “Obtention de programmes corrects par raffinement dans un langage de haut niveau”, Université Paris-Saclay & Université Paris-Sud, December 13, 2016, supervised by J.-C. Filliâtre and A. Paskevich.

PhD in progress: M. Clochard, “A unique language for developing programs and prove them at the same time”, since Oct. 2013, supervised by C. Marché and A. Paskevich.

PhD in progress: D. Declerck, “Vérification par des techniques de test et model checking de programmes C11”, since Sep. 2014, supervised by F. Zaïdi (LRI) and S. Conchon.

PhD in progress: M. Roux, “Model Checking de systèmes paramétrés et temporisés”, since Sep. 2015, supervised by Sylvain Conchon.

PhD in progress: M. Pereira, “A Verified Graph Library. Tools and techniques for the verification of modular higher-order programs, with extraction”, since May 2015, supervised by J.-C. Filliâtre.

PhD in progress: A. Coquereau, “[ErgoFast] Amélioration de performances pour le solveur SMT Alt-Ergo : conception d’outils d’analyse, optimisations et structures de données efficaces pour OCaml”, since Sep. 2015, supervised by Sylvain Conchon, Fabrice Le Fessant et Michel Mauny.

PhD in progress: F. Faissole, “Stabilité(s): liens entre l’arithmétique flottante et l’analyse numérique”, since Oct 2016, supervised by S. Boldo and A. Chapoutot.

10.2.4. Juries

J.-C. Filliâtre: president of the PhD committee of A. Djoudi, “Analyse statique au niveau binaire”, Université Paris Saclay, France, December 2016.

S. Conchon: examiner of the PhD of M. Morterol,“Méthodes avancées de raisonnement en logique propositionnelle : application aux réseaux métaboliques”, Université Paris-Saclay, December 2016.

S. Conchon: reviewer of the PhD of A. Blanchard, “Aide à la vérification de programmes concurrents par transformation de code et de spécifications”, Université d’Orléans, December 2016.

S. Conchon: reviewer of the HDR of X. Thirioux, “Verifying Embedded Systems”, Institut National Polytechnique de Toulouse, September 2016.

S. Conchon : examiner of the PhD of L. Cabaret, “Algorithmes d’étiquetage en composantes connexes efficaces pour architectures hautes performances”, September 2016.

10.3. Popularization

- A. Charguéraud is one of the three organizers of the *Concours Castor informatique* http://castor-informatique.fr/. The purpose of the Concours Castor in to introduce pupils (from *CM1* to *Terminale*) to computer sciences. 475,000 teenagers played with the interactive exercises in November 2016.
- S. Boldo is a speaker for a MOOC for computer science teachers. She was also invited to Poitiers in November 2016 to discuss with teachers and present this MOOC.
- S. Boldo was invited to a panel about teaching computer science before university in Besançon in June 2016 during the GDR GPL days.
- During the "Fête de la science" on October 14th to 16th, S. Boldo gave several talks about computer arithmetic to teenagers and F. Faissole run a stand about an introduction to programming with robots.
- S. Boldo and F. Voisin did an introduction to computer science with an activity on computer hardware as a 1-hour extracurricular activity in schools for pupils in *CM1-CM2* on October 4th.
- S. Boldo gave a talk during a girls & maths weekend on November 22nd. See http://www.animath.fr/spip.php?article2897&lang=fr.

11. Bibliography

Major publications by the team in recent years

[1] J. C. BLANCHETTE, A. PASKEVICH. *TFF1: The TPTP typed first-order form with rank-1 polymorphism*, in "24th International Conference on Automated Deduction (CADE-24)", Lake Placid, USA, Lecture Notes in Artificial Intelligence, Springer, June 2013, vol. 7898, http://hal.inria.fr/hal-00825086

[2] F. BOBOT, S. CONCHON, É. CONTEJEAN, M. IGUERNELALA, A. MAHBOUBI, A. MEBSOUT, G. MELQUIOND. *A Simplex-Based Extension of Fourier-Motzkin for Solving Linear Integer Arithmetic*, in "IJCAR 2012: Proceedings of the 6th International Joint Conference on Automated Reasoning", Manchester, UK, B. GRAMLICH, D. MILLER, U. SATTLER (editors), Lecture Notes in Computer Science, Springer, June 2012, vol. 7364, pp. 67–81, http://hal.inria.fr/hal-00687640

[3] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, A. PASKEVICH. *Let's Verify This with Why3*, in "International Journal on Software Tools for Technology Transfer (STTT)", 2015, vol. 17, n° 6, pp. 709–727, http://hal.inria.fr/hal-00967132/en

[4] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Wave Equation Numerical Resolution: a Comprehensive Mechanized Proof of a C Program*, in "Journal of Automated Reasoning", April 2013, vol. 50, n° 4, pp. 423–456, http://hal.inria.fr/hal-00649240/en/

[5] S. BOLDO, G. MELQUIOND. *Flocq: A Unified Library for Proving Floating-point Algorithms in Coq*, in "Proceedings of the 20th IEEE Symposium on Computer Arithmetic", Tübingen, Germany, E. ANTELO, D. HOUGH, P. IENNE (editors), 2011, pp. 243–252, http://hal.archives-ouvertes.fr/inria-00534854/

[6] A. CHARGUÉRAUD. *Pretty-Big-Step Semantics*, in "Proceedings of the 22nd European Symposium on Programming", M. FELLEISEN, P. GARDNER (editors), Lecture Notes in Computer Science, Springer, March 2013, vol. 7792, pp. 41–60, http://hal.inria.fr/hal-00798227

[7] S. CONCHON, A. GOEL, S. KRSTIĆ, A. MEBSOUT, F. ZAÏDI. *Cubicle: A Parallel SMT-based Model Checker for Parameterized Systems*, in "CAV 2012: Proceedings of the 24th International Conference on Computer Aided Verification", Berkeley, California, USA, M. PARTHASARATHY, S. A. SESHIA (editors), Lecture Notes in Computer Science, Springer, July 2012, vol. 7358, http://hal.archives-ouvertes.fr/hal-00799272

[8] J.-C. FILLIÂTRE, L. GONDELMAN, A. PASKEVICH. *The Spirit of Ghost Code*, in "26th International Conference on Computer Aided Verification", Vienna, Austria, A. BIERE, R. BLOEM (editors), Lecture Notes in Computer Science, Springer, July 2014, vol. 8859, pp. 1–16, http://hal.archives-ouvertes.fr/hal-00873187/en/

[9] C. MARCHÉ. *Verification of the Functional Behavior of a Floating-Point Program: an Industrial Case Study*, in "Science of Computer Programming", March 2014, vol. 96, n° 3, pp. 279–296, http://hal.inria.fr/hal-00967124/en

[10] G. MELQUIOND. *Proving bounds on real-valued functions with computations*, in "Proceedings of the 4th International Joint Conference on Automated Reasoning", Sydney, Australia, A. ARMANDO, P. BAUMGARTNER, G. DOWEK (editors), Lecture Notes in Artificial Intelligence, 2008, vol. 5195, pp. 2–17

Publications of the year

Articles in International Peer-Reviewed Journals

[11] U. A. ACAR, A. CHARGUÉRAUD, M. RAINEY. *Oracle-Guided Scheduling for Controlling Granularity in Implicitly Parallel Languages*, in "Journal of Functional Programming", November 2016, vol. 26 [*DOI :* 10.1017/S0956796816000101], https://hal.inria.fr/hal-01409069

[12] S. BOLDO, C. LELAY, G. MELQUIOND. *Formalization of Real Analysis: A Survey of Proof Assistants and Libraries*, in "Mathematical Structures in Computer Science", October 2016, vol. 26, n° 7, pp. 1196-1233 [*DOI :* 10.1017/S0960129514000437], https://hal.inria.fr/hal-00806920

[13] C. DROSS, S. CONCHON, J. KANIG, A. PASKEVICH. *Adding Decision Procedures to SMT Solvers using Axioms with Triggers*, in "Journal of Automated Reasoning", 2016, vol. 56, n° 4, pp. 387-457 [*DOI :* 10.1007/s10817-015-9352-2], https://hal.archives-ouvertes.fr/hal-01221066

[14] J.-C. FILLIÂTRE, L. GONDELMAN, A. PASKEVICH. *The Spirit of Ghost Code*, in "Formal Methods in System Design", June 2016, vol. 48, n° 3, pp. 152-174, Extended version of https://hal.inria.fr/hal-00873187 [*DOI :* 10.1007/s10703-016-0243-x], https://hal.archives-ouvertes.fr/hal-01396864

[15] É. MARTIN-DOREL, G. MELQUIOND. *Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq*, in "Journal of Automated Reasoning", October 2016, vol. 57, n° 3, pp. 187-217 [*DOI :* 10.1007/s10817-015-9350-4], https://hal.inria.fr/hal-01086460

International Conferences with Proceedings

[16] U. A. ACAR, A. CHARGUÉRAUD, M. RAINEY, F. SIECZKOWSKI. *Dag-calculus: a calculus for parallel computation*, in "Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming (ICFP)", Nara, Japan, September 2016, pp. 18 - 32 [*DOI :* 10.1145/2951913.2951946], https://hal.inria.fr/hal-01409022

[17] S. BOLDO. *Computing a correct and tight rounding error bound using rounding-to-nearest*, in "9th International Workshop on Numerical Software Verification", Toronto, Canada, July 2016, https://hal.inria.fr/hal-01377152

[18] S. BOLDO, F. CLÉMENT, F. FAISSOLE, V. MARTIN, M. MAYERO. *A Coq Formal Proof of the Lax–Milgram theorem*, in "6th ACM SIGPLAN Conference on Certified Programs and Proofs", Paris, France, January 2017, https://hal.inria.fr/hal-01391578

[19] A. CHARGUÉRAUD. *Higher-order representation predicates in separation logic*, in "Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP)", St. Petersburg, FL, United States, ACM, January 2016, pp. 3 - 14 [*DOI :* 10.1145/2854065.2854068], https://hal.inria.fr/hal-01408670

[20] A. CHARGUÉRAUD, F. POTTIER. *Temporary Read-Only Permissions for Separation Logic*, in "Proceedings of the 26th European Symposium on Programming (ESOP 2017)", Uppsala, Sweden, April 2017, https://hal.inria.fr/hal-01408657

[21] M. CLOCHARD, L. GONDELMAN, M. PEREIRA. *The Matrix Reproved (Verification Pearl)*, in "VSTTE 2016", Toronto, Canada, July 2016, https://hal.inria.fr/hal-01316902

[22] J.-C. FILLIÂTRE, M. PEREIRA. *A Modular Way to Reason About Iteration*, in "8th NASA Formal Methods Symposium", Minneapolis, United States, June 2016, https://hal.inria.fr/hal-01281759

[23] J.-C. FILLIÂTRE, M. PEREIRA. *Producing All Ideals of a Forest, Formally (Verification Pearl)*, in "VSTTE 2016", Toronto, Canada, July 2016, pp. 46 - 55 [*DOI :* 10.1007/978-3-319-48869-1_4], https://hal.inria.fr/hal-01316859

[24] C. FUMEX, C. DROSS, J. GERLACH, C. MARCHÉ. *Specification and Proof of High-Level Functional Properties of Bit-Level Programs*, in "NASA Formal methods", Minneapolis, United States, NASA Formal methods, Springer, June 2016, https://hal.inria.fr/hal-01314876

[25] D. HAUZAR, C. MARCHÉ, Y. MOY. *Counterexamples from Proof Failures in SPARK*, in "Software Engineering and Formal Methods", Vienna, Austria, Software Engineering and Formal Methods, Springer, July 2016, https://hal.inria.fr/hal-01314885

[26] N. KOSMATOV, C. MARCHÉ, Y. MOY, J. SIGNOLES. *Static versus Dynamic Verification in Why3, Frama-C and SPARK 2014*, in "7th International Symposium on Leveraging Applications", Corfu, Greece, 7th International Symposium on Leveraging Applications, Springer, October 2016, 16 p. , https://hal.inria.fr/hal-01344110

[27] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, in "Interactive Theorem Proving", Nancy, France, J. C. BLANCHETTE, S. MERZ (editors), Lecture Notes in Computer Science, August 2016, vol. 9807 [*DOI :* 10.1007/978-3-319-43144-4_17], https://hal.inria.fr/hal-01289616

National Conferences with Proceedings

[28] R. CHEN, J.-J. LÉVY. *Une preuve formelle de l'algorithme de Tarjan-1972 pour trouver les composantes fortement connexes dans un graphe*, in "JFLA 2017 - Vingt-huitièmes Journées Francophones des Langages Applicatifs", Gourette, France, Vingt-huitièmes Journées Francophones des Langages Applicatifs, January 2017, https://hal.inria.fr/hal-01422215

[29] M. CLOCHARD. *Preuves taillées en biseau*, in "vingt-huitièmes Journées Francophones des Langages Applicatifs (JFLA)", Gourette, France, January 2017, https://hal.inria.fr/hal-01404935

[30] M. PEREIRA. *Défonctionnaliser pour prouver*, in "JFLA 2017", Gourette, France, January 2017, https://hal.inria.fr/hal-01378068

Conferences without Proceedings

[31] S. BOLDO. *Iterators: where folds fail*, in "Workshop on High-Consequence Control Verification", Toronto, Canada, July 2016, https://hal.inria.fr/hal-01377155

[32] F. FAISSOLE, B. SPITTERS. *Synthetic topology in HoTT for probabilistic programming*, in "The Third International Workshop on Coq for Programming Languages (CoqPL 2017)", Paris, France, January 2017, https://hal.inria.fr/hal-01405762

[33] J.-C. FILLIÂTRE, M. PEREIRA. *Itérer avec confiance*, in "Journées Francophones des Langages Applicatifs", Saint-Malo, France, January 2016, https://hal.inria.fr/hal-01240891

Research Reports

[34] R. CHEN, M. CLOCHARD, C. MARCHÉ. *A Formal Proof of a Unix Path Resolution Algorithm*, Inria, December 2016, n° RR-8987, 27 p. , https://hal.inria.fr/hal-01406848

[35] D. HAUZAR, C. MARCHÉ, Y. MOY. *Counterexamples from proof failures in the SPARK program verifier*, Inria, February 2016, n° RR-8854, 22 p. , https://hal.inria.fr/hal-01271174

Other Publications

[36] S. BOLDO, S. GRAILLAT, J.-M. MULLER. *On the robustness of the 2Sum and Fast2Sum algorithms*, May 2016, working paper or preprint, https://hal-ens-lyon.archives-ouvertes.fr/ensl-01310023

[37] J.-C. FILLIÂTRE, L. GONDELMAN, A. PASKEVICH. *A Pragmatic Type System for Deductive Verification*, February 2016, working paper or preprint, https://hal.inria.fr/hal-01256434

References in notes

[38] B. BECKERT, R. HÄHNLE, P. H. SCHMITT (editors). *Verification of Object-Oriented Software: The KeY Approach*, Lecture Notes in Computer Science, Springer, 2007, vol. 4334

[39] U. A. ACAR, A. CHARGUÉRAUD, M. RAINEY. *Theory and Practice of Chunked Sequences*, in "European Symposium on Algorithms", Wroclaw, Poland, A. SCHULZ, D. WAGNER (editors), Springer, September 2014, vol. Lecture Notes in Computer Science, n° 8737, pp. 25–36, https://hal.inria.fr/hal-01087245

[40] J. B. ALMEIDA, M. BARBOSA, J.-C. FILLIÂTRE, J. S. PINTO, B. VIEIRA. *CAOVerif: An Open-Source Deductive Verification Platform for Cryptographic Software Implementations*, in "Science of Computer Programming", October 2012

[41] A. AYAD, C. MARCHÉ. *Multi-Prover Verification of Floating-Point Programs*, in "Fifth International Joint Conference on Automated Reasoning", Edinburgh, Scotland, J. GIESL, R. HÄHNLE (editors), Lecture Notes in Artificial Intelligence, Springer, July 2010, vol. 6173, pp. 127–141, http://hal.inria.fr/inria-00534333

[42] D. BAELDE, P. COURTIEU, D. GROSS-AMBLARD, C. PAULIN-MOHRING. *Towards Provably Robust Watermarking*, in "ITP 2012", Lecture Notes in Computer Science, August 2012, vol. 7406, http://hal.inria.fr/hal-00682398

[43] C. BARRETT, C. TINELLI. *CVC3*, in "19th International Conference on Computer Aided Verification", Berlin, Germany, W. DAMM, H. HERMANNS (editors), Lecture Notes in Computer Science, Springer, July 2007, vol. 4590, pp. 298–302

[44] P. BAUDIN, J.-C. FILLIÂTRE, C. MARCHÉ, B. MONATE, Y. MOY, V. PREVOSTO. *ACSL: ANSI/ISO C Specification Language, version 1.4*, 2009

[45] P. BEHM, P. BENOIT, A. FAIVRE, J.-M. MEYNADIER. *METEOR : A successful application of B in a large project*, in "Proceedings of FM'99: World Congress on Formal Methods", J. M. WING, J. WOODCOCK, J. DAVIES (editors), Lecture Notes in Computer Science (Springer-Verlag), Springer Verlag, September 1999, pp. 369–387

[46] F. BOBOT, S. CONCHON, É. CONTEJEAN, M. IGUERNELALA, S. LESCUYER, A. MEBSOUT. *The Alt-Ergo Automated Theorem Prover*, 2008

[47] F. BOBOT, J.-C. FILLIÂTRE. *Separation Predicates: a Taste of Separation Logic in First-Order Logic*, in "14th International Conference on Formal Ingineering Methods (ICFEM)", Kyoto, Japan, Lecture Notes in Computer Science, Springer, November 2012, vol. 7635, http://hal.inria.fr/hal-00825088

[48] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, G. MELQUIOND, A. PASKEVICH. *Preserving User Proofs Across Specification Changes*, in "Verified Software: Theories, Tools, Experiments (5th International Conference VSTTE)", Atherton, USA, E. COHEN, A. RYBALCHENKO (editors), Lecture Notes in Computer Science, Springer, May 2013, vol. 8164, pp. 191–201, http://hal.inria.fr/hal-00875395

[49] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, G. MELQUIOND, A. PASKEVICH. *The Why3 platform, version 0.81*, version 0.81, LRI, CNRS & Univ. Paris-Sud & Inria Saclay, March 2013, http://hal.inria.fr/hal-00822856/

[50] M. BODIN, A. CHARGUÉRAUD, D. FILARETTI, P. GARDNER, S. MAFFEIS, D. NAUDZIUNIENE, A. SCHMITT, G. SMITH. *A Trusted Mechanised JavaScript Specification*, in "Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", San Diego, USA, ACM Press, January 2014, http://hal.inria.fr/hal-00910135

[51] S. BOLDO. *How to Compute the Area of a Triangle: a Formal Revisit*, in "Proceedings of the 21th IEEE Symposium on Computer Arithmetic", Austin, Texas, USA, 2013, http://hal.inria.fr/hal-00790071

[52] S. BOLDO. *Deductive Formal Verification: How To Make Your Floating-Point Programs Behave*, Université Paris-Sud, October 2014, Thèse d'habilitation, https://hal.inria.fr/tel-01089643

[53] S. BOLDO. *Formal verification of tricky numerical computations*, in "16th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics", Würzburg, Germany, September 2014, https://hal.inria.fr/hal-01088692

[54] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Formal Proof of a Wave Equation Resolution Scheme: the Method Error*, in "Proceedings of the First Interactive Theorem

Proving Conference", Edinburgh, Scotland, M. KAUFMANN, L. C. PAULSON (editors), LNCS, Springer, July 2010, vol. 6172, pp. 147–162, http://hal.inria.fr/inria-00450789/

[55] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Trusting Computations: a Mechanized Proof from Partial Differential Equations to Actual Program*, in "Computers and Mathematics with Applications", 2014, vol. 68, n° 3, pp. 325–352, http://hal.inria.fr/hal-00769201

[56] S. BOLDO, J.-C. FILLIÂTRE, G. MELQUIOND. *Combining Coq and Gappa for Certifying Floating-Point Programs*, in "16th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning", Grand Bend, Canada, Lecture Notes in Artificial Intelligence, Springer, July 2009, vol. 5625, pp. 59–74

[57] S. BOLDO, J.-H. JOURDAN, X. LEROY, G. MELQUIOND. *A Formally-Verified C Compiler Supporting Floating-Point Arithmetic*, in "Proceedings of the 21th IEEE Symposium on Computer Arithmetic", Austin, Texas, USA, 2013, http://hal.inria.fr/hal-00743090

[58] S. BOLDO, J.-H. JOURDAN, X. LEROY, G. MELQUIOND. *Verified Compilation of Floating-Point Computations*, in "Journal of Automated Reasoning", February 2015, vol. 54, n° 2, pp. 135-163, https://hal.inria.fr/hal-00862689

[59] S. BOLDO, C. LELAY, G. MELQUIOND. *Improving Real Analysis in Coq: a User-Friendly Approach to Integrals and Derivatives*, in "Proceedings of the Second International Conference on Certified Programs and Proofs", Kyoto, Japan, C. HAWBLITZEL, D. MILLER (editors), Lecture Notes in Computer Science, December 2012, vol. 7679, pp. 289–304, http://hal.inria.fr/hal-00712938

[60] S. BOLDO, C. LELAY, G. MELQUIOND. *Coquelicot: A User-Friendly Library of Real Analysis for Coq*, in "Mathematics in Computer Science", June 2015, vol. 9, n° 1, pp. 41-62, http://hal.inria.fr/hal-00860648

[61] S. BOLDO, C. MARCHÉ. *Formal verification of numerical programs: from C annotated programs to mechanical proofs*, in "Mathematics in Computer Science", 2011, vol. 5, pp. 377–393, http://hal.inria.fr/hal-00777605

[62] S. BOLDO, T. M. T. NGUYEN. *Proofs of numerical programs when the compiler optimizes*, in "Innovations in Systems and Software Engineering", 2011, vol. 7, pp. 151–160, http://hal.inria.fr/hal-00777639

[63] T. BORMER, M. BROCKSCHMIDT, D. DISTEFANO, G. ERNST, J.-C. FILLIÂTRE, R. GRIGORE, M. HUISMAN, V. KLEBANOV, C. MARCHÉ, R. MONAHAN, W. MOSTOWSKI, N. POLIKARPOVA, C. SCHEBEN, G. SCHELLHORN, B. TOFAN, J. TSCHANNEN, M. ULBRICH. *The COST IC0701 Verification Competition 2011*, in "Formal Verification of Object-Oriented Software, Revised Selected Papers Presented at the International Conference, FoVeOOS 2011", B. BECKERT, F. DAMIANI, D. GUROV (editors), Lecture Notes in Computer Science, Springer, 2012, vol. 7421, http://hal.inria.fr/hal-00789525

[64] L. BURDY, Y. CHEON, D. R. COK, M. D. ERNST, J. R. KINIRY, G. T. LEAVENS, K. R. M. LEINO, E. POLL. *An overview of JML tools and applications*, in "International Journal on Software Tools for Technology Transfer (STTT)", June 2005, vol. 7, n° 3, pp. 212–232

[65] R. CHAPMAN, F. SCHANDA. *Are We There Yet? 20 Years of Industrial Theorem Proving with SPARK*, in "Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings", G. KLEIN, R. GAMBOA (editors), Lecture Notes in Computer Science, Springer, 2014, vol. 8558, pp. 17–26

[66] M. CLOCHARD. *Automatically verified implementation of data structures based on AVL trees*, in "6th Working Conference on Verified Software: Theories, Tools and Experiments (VSTTE)", Vienna, Austria, D. GIANNAKOPOULOU, D. KROENING (editors), Lecture Notes in Computer Science, Springer, July 2014, vol. 8471, pp. 167–180, http://hal.inria.fr/hal-01067217

[67] M. CLOCHARD, J.-C. FILLIÂTRE, C. MARCHÉ, A. PASKEVICH. *Formalizing Semantics with an Automatic Program Verifier*, in "6th Working Conference on Verified Software: Theories, Tools and Experiments (VSTTE)", Vienna, Austria, D. GIANNAKOPOULOU, D. KROENING (editors), Lecture Notes in Computer Science, Springer, July 2014, vol. 8471, pp. 37–51, http://hal.inria.fr/hal-01067197

[68] M. CLOCHARD, L. GONDELMAN. *Double WP: vers une preuve automatique d'un compilateur*, in "Vingt-sixièmes Journées Francophones des Langages Applicatifs", Val d'Ajol, France, January 2015, https://hal.inria.fr/hal-01094488

[69] M. CLOCHARD, C. MARCHÉ, A. PASKEVICH. *Verified Programs with Binders*, in "Programming Languages meets Program Verification (PLPV)", ACM Press, 2014, http://hal.inria.fr/hal-00913431

[70] S. CONCHON. *SMT Techniques and their Applications: from Alt-Ergo to Cubicle*, Université Paris-Sud, December 2012, In English, http://www.lri.fr/~conchon/publis/conchonHDR.pdf, Thèse d'habilitation

[71] S. CONCHON, É. CONTEJEAN, M. IGUERNELALA. *Canonized Rewriting and Ground AC Completion Modulo Shostak Theories*, in "Tools and Algorithms for the Construction and Analysis of Systems", Saarbrücken, Germany, P. A. ABDULLA, K. R. M. LEINO (editors), Lecture Notes in Computer Science, Springer, April 2011, vol. 6605, pp. 45-59, http://hal.inria.fr/hal-00777663

[72] S. CONCHON, É. CONTEJEAN, M. IGUERNELALA. *Canonized Rewriting and Ground AC Completion Modulo Shostak Theories : Design and Implementation*, in "Logical Methods in Computer Science", September 2012, vol. 8, n^{o} 3, pp. 1–29, hal.inria.fr/hal-00798082

[73] S. CONCHON, D. DECLERCK, L. MARANGET, A. MEBSOUT. *Vérification de programmes C concurrents avec Cubicle : Enfoncer les barrières*, in "Vingt-cinquièmes Journées Francophones des Langages Applicatifs", Fréjus, France, January 2014, https://hal.inria.fr/hal-01088655

[74] S. CONCHON, A. GOEL, S. KRSTIĆ, A. MEBSOUT, F. ZAÏDI. *Invariants for Finite Instances and Beyond*, in "FMCAD", Portland, Oregon, États-Unis, October 2013, pp. 61–68, http://hal.archives-ouvertes.fr/hal-00924640

[75] S. CONCHON, M. IGUERNELALA. *Tuning the Alt-Ergo SMT Solver for B Proof Obligations*, in "Abstract State Machines, Alloy, B, VDM, and Z (ABZ)", Toulouse, France, Lecture Notes in Computer Science, Springer, June 2014, vol. 8477, pp. 294–297, https://hal.inria.fr/hal-01093000

[76] S. CONCHON, M. IGUERNELALA, A. MEBSOUT. *A Collaborative Framework for Non-Linear Integer Arithmetic Reasoning in Alt-Ergo*, 2013, https://hal.archives-ouvertes.fr/hal-00924646

[77] S. CONCHON, A. MEBSOUT, F. ZAÏDI. *Vérification de systèmes paramétrés avec Cubicle*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013, http://hal.inria.fr/hal-00778832

[78] S. CONCHON, G. MELQUIOND, C. ROUX, M. IGUERNELALA. *Built-in Treatment of an Axiomatic Floating-Point Theory for SMT Solvers*, in "SMT workshop", Manchester, UK, P. FONTAINE, A. GOEL (editors), LORIA, 2012, pp. 12–21

[79] M. DAHLWEID, M. MOSKAL, T. SANTEN, S. TOBIES, W. SCHULTE. *VCC: Contract-based modular verification of concurrent C*, in "31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Companion Volume", IEEE Comp. Soc. Press, 2009, pp. 429-430

[80] D. DELAHAYE, C. DUBOIS, C. MARCHÉ, D. MENTRÉ. *The BWare Project: Building a Proof Platform for the Automated Verification of B Proof Obligations*, in "Abstract State Machines, Alloy, B, VDM, and Z (ABZ)", Toulouse, France, Lecture Notes in Computer Science, Springer, June 2014, vol. 8477, pp. 290–293, http://hal.inria.fr/hal-00998092/en/

[81] D. DELAHAYE, C. MARCHÉ, D. MENTRÉ. *Le projet BWare : une plate-forme pour la vérification automatique d'obligations de preuve B*, in "Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL)", Paris, France, EasyChair, June 2014, http://hal.inria.fr/hal-00998094/en/

[82] C. DROSS, S. CONCHON, J. KANIG, A. PASKEVICH. *Reasoning with Triggers*, in "SMT workshop", Manchester, UK, P. FONTAINE, A. GOEL (editors), LORIA, 2012

[83] C. DROSS. *Generic Decision Procedures for Axiomatic First-Order Theories*, Université Paris-Sud, April 2014, http://tel.archives-ouvertes.fr/tel-01002190

[84] J.-C. FILLIÂTRE. *Combining Interactive and Automated Theorem Proving in Why3 (invited talk)*, in "Automation in Proof Assistants 2012", Tallinn, Estonia, K. HELJANKO, H. HERBELIN (editors), April 2012

[85] J.-C. FILLIÂTRE. *Combining Interactive and Automated Theorem Proving using Why3 (invited tutorial)*, in "Second International Workshop on Intermediate Verification Languages (BOOGIE 2012)", Berkeley, California, USA, Z. RAKAMARIĆ (editor), July 2012

[86] J.-C. FILLIÂTRE. *Verifying Two Lines of C with Why3: an Exercise in Program Verification*, in "Verified Software: Theories, Tools, Experiments (4th International Conference VSTTE)", Philadelphia, USA, R. JOSHI, P. MÜLLER, A. PODELSKI (editors), Lecture Notes in Computer Science, Springer, January 2012, vol. 7152, pp. 83–97

[87] J.-C. FILLIÂTRE. *Deductive Program Verification*, in "Programming Languages Mentoring Workshop (PLMW 2013)", Rome, Italy, N. FOSTER, P. GARDNER, A. SCHMITT, G. SMITH, P. THIEMAN, T. WRIGSTAD (editors), January 2013, http://hal.inria.fr/hal-00799190

[88] J.-C. FILLIÂTRE. *One Logic To Use Them All*, in "24th International Conference on Automated Deduction (CADE-24)", Lake Placid, USA, Lecture Notes in Artificial Intelligence, Springer, June 2013, vol. 7898, pp. 1–20, http://hal.inria.fr/hal-00809651/en/

[89] J.-C. FILLIÂTRE, A. PASKEVICH. *Why3 — Where Programs Meet Provers*, in "Proceedings of the 22nd European Symposium on Programming", M. FELLEISEN, P. GARDNER (editors), Lecture Notes in Computer Science, Springer, March 2013, vol. 7792, pp. 125–128, http://hal.inria.fr/hal-00789533

[90] J.-C. FILLIÂTRE, A. PASKEVICH, A. STUMP. *The 2nd Verified Software Competition: Experience Report*, in "COMPARE2012: 1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems", Manchester, UK, V. KLEBANOV, S. GREBING (editors), EasyChair, June 2012, http://hal.inria.fr/hal-00798777

[91] P. HERMS. *Certification of a Tool Chain for Deductive Program Verification*, Université Paris-Sud, January 2013, http://tel.archives-ouvertes.fr/tel-00789543

[92] P. HERMS, C. MARCHÉ, B. MONATE. *A Certified Multi-prover Verification Condition Generator*, in "Verified Software: Theories, Tools, Experiments (4th International Conference VSTTE)", Philadelphia, USA, R. JOSHI, P. MÜLLER, A. PODELSKI (editors), Lecture Notes in Computer Science, Springer, January 2012, vol. 7152, pp. 2–17, http://hal.inria.fr/hal-00639977

[93] M. IGUERNELALA. *Strengthening the Heart of an SMT-Solver: Design and Implementation of Efficient Decision Procedures*, Université Paris-Sud, June 2013, http://tel.archives-ouvertes.fr/tel-00842555

[94] D. ISHII, G. MELQUIOND, S. NAKAJIMA. *Inductive Verification of Hybrid Automata with Strongest Post-condition Calculus*, in "Proceedings of the 10th Conference on Integrated Formal Methods", Turku, Finland, E. B. JOHNSEN, L. PETRE (editors), Lecture Notes in Computer Science, 2013, vol. 7940, pp. 139–153, http://hal.inria.fr/hal-00806701

[95] J. KANIG, E. SCHONBERG, C. DROSS. *Hi-Lite: the convergence of compiler technology and program verification*, in "Proceedings of the 2012 ACM Conference on High Integrity Language Technology, HILT '12", Boston, USA, B. BROSGOL, J. BOLENG, S. T. TAFT (editors), ACM Press, 2012, pp. 27–34

[96] G. KLEIN, J. ANDRONICK, K. ELPHINSTONE, G. HEISER, D. COCK, P. DERRIN, D. ELKADUWE, K. ENGELHARDT, R. KOLANSKI, M. NORRISH, T. SEWELL, H. TUCH, S. WINWOOD. *seL4: Formal verification of an OS kernel*, in "Communications of the ACM", June 2010, vol. 53, n^{o} 6, pp. 107–115

[97] C. LELAY. *A New Formalization of Power Series in Coq*, in "5th Coq Workshop", Rennes, France, July 2013, pp. 1–2, http://hal.inria.fr/hal-00880212

[98] C. LELAY. *Coq passe le bac*, in "JFLA - Journées francophones des langages applicatifs", Fréjus, France, January 2014

[99] C. LELAY, G. MELQUIOND. *Différentiabilité et intégrabilité en Coq. Application à la formule de d'Alembert*, in "Vingt-troisièmes Journées Francophones des Langages Applicatifs", Carnac, France, February 2012, http://hal.inria.fr/hal-00642206/fr/

[100] X. LEROY. *A formally verified compiler back-end*, in "Journal of Automated Reasoning", 2009, vol. 43, n^{o} 4, pp. 363–446, http://hal.inria.fr/inria-00360768/en/

[101] C. B. LOURENÇO, S. LAMRAOUI, S. NAKAJIMA, J. S. PINTO. *Studying Verification Conditions for Imperative Programs*, in "Electronic Communication of the European Association of Software Science and Technology", 2015, vol. 72

[102] C. MARCHÉ, A. TAFAT. *Weakest Precondition Calculus, revisited using Why3*, Inria, December 2012, n^{o} RR-8185, http://hal.inria.fr/hal-00766171

[103] C. MARCHÉ, A. TAFAT. *Calcul de plus faible précondition, revisité en Why3*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013, http://hal.inria.fr/hal-00778791

[104] C. MARCHÉ. *Verification of the Functional Behavior of a Floating-Point Program: an Industrial Case Study*, in "Science of Computer Programming", March 2014, vol. 96, n° 3, pp. 279–296, http://hal.inria.fr/hal-00967124/en

[105] É. MARTIN-DOREL, G. MELQUIOND, J.-M. MULLER. *Some Issues related to Double Roundings*, in "BIT Numerical Mathematics", 2013, vol. 53, n° 4, pp. 897–924, http://hal-ens-lyon.archives-ouvertes.fr/ensl-00644408

[106] A. MEBSOUT. *Invariants inference for model checking of parameterized systems*, Université Paris-Sud, September 2014, https://tel.archives-ouvertes.fr/tel-01073980

[107] G. MELQUIOND. *Floating-point arithmetic in the Coq system*, in "Information and Computation", 2012, vol. 216, pp. 14–23, http://hal.inria.fr/hal-00797913

[108] G. MELQUIOND, W. G. NOWAK, P. ZIMMERMANN. *Numerical Approximation of the Masser-Gramain Constant to Four Decimal Digits: delta=1.819...*, in "Mathematics of Computation", 2013, vol. 82, pp. 1235–1246, http://hal.inria.fr/hal-00644166/en/

[109] D. MENTRÉ, C. MARCHÉ, J.-C. FILLIÂTRE, M. ASUKA. *Discharging Proof Obligations from Atelier B using Multiple Automated Provers*, in "ABZ'2012 - 3rd International Conference on Abstract State Machines, Alloy, B and Z", Pisa, Italy, S. REEVES, E. RICCOBENE (editors), Lecture Notes in Computer Science, Springer, June 2012, vol. 7316, pp. 238–251, http://hal.inria.fr/hal-00681781/en/

[110] J.-M. MULLER, N. BRISEBARRE, F. DE DINECHIN, C.-P. JEANNEROD, V. LEFÈVRE, G. MELQUIOND, N. REVOL, D. STEHLÉ, S. TORRES. *Handbook of Floating-Point Arithmetic*, Birkhäuser, 2010

[111] T. M. T. NGUYEN, C. MARCHÉ. *Hardware-Dependent Proofs of Numerical Programs*, in "Certified Programs and Proofs", J.-P. JOUANNAUD, Z. SHAO (editors), Lecture Notes in Computer Science, Springer, December 2011, pp. 314–329, http://hal.inria.fr/hal-00772508

[112] T. M. T. NGUYEN. *Taking architecture and compiler into account in formal proofs of numerical programs*, Université Paris-Sud, June 2012, http://tel.archives-ouvertes.fr/tel-00710193

[113] M. NORRISH. *C Formalised in HOL*, University of Cambridge, November 1998

[114] M. PEREIRA, J.-C. FILLIÂTRE, S. M. DE SOUSA. *ARMY: a Deductive Verification Platform for ARM Programs Using Why3*, in "INForum 2012", September 2012

[115] P. ROUX. *Formal Proofs of Rounding Error Bounds*, in "Journal of Automated Reasoning", 2015, https://hal.archives-ouvertes.fr/hal-01091189

[116] N. SCHIRMER. *Verification of Sequential Imperative Programs in Isabelle/HOL*, Technische Universität München, 2006

[117] A. TAFAT. *Preuves par raffinement de programmes avec pointeurs*, Université Paris-Sud, September 2013, http://tel.archives-ouvertes.fr/tel-00874679

[118] F. DE DINECHIN, C. LAUTER, G. MELQUIOND. *Certifying the floating-point implementation of an elementary function using Gappa*, in "IEEE Transactions on Computers", 2011, vol. 60, n^{o} 2, pp. 242–253, http://hal.inria.fr/inria-00533968/en/

[119] L. DE MOURA, N. BJØRNER. *Z3, An Efficient SMT Solver*, in "TACAS", Lecture Notes in Computer Science, Springer, 2008, vol. 4963, pp. 337–340