IN PARTNERSHIP WITH:
CNRS

Université Paris-Sud (Paris 11)

Activity Report 2019

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

The Toccata team (http://toccata.lri.fr/) is a research team common to Inria Saclay–Île-de-France, CNRS, and Université Paris-Sud. Team members are also members of the larger VALS research group (Verification of Algorithms, Languages and systems; http://vals.lri.fr/) of the LRI (Laboratoire de Recherche en Informatique, UMR 8623).

Keywords:

Computer Science and Digital Science:

A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.6. - Concurrent programming
A2.1.10. - Domain-specific languages
A2.1.11. - Proof languages
A2.4.2. - Model-checking
A2.4.3. - Proofs
A6.2.1. - Numerical analysis of PDE and ODE
A7.2. - Logic in Computer Science
A7.2.1. - Decision procedures
A7.2.2. - Automated Theorem Proving
A7.2.3. - Interactive Theorem Proving
A7.2.4. - Mechanized Formalization of Mathematics
A8.10. - Computer arithmetic

Other Research Topics and Application Domains:

B5.2.2. - Railway
B5.2.3. - Aviation
B5.2.4. - Aerospace
B6.1. - Software industry
B9.5.1. - Computer science
B9.5.2. - Mathematics

1. Team, Visitors, External Collaborators

Research Scientists

Claude Marché [Team leader, Inria, Senior Researcher, HDR]
Sylvie Boldo [Inria, Senior Researcher, HDR]
Jean-Christophe Filliâtre [CNRS, Senior Researcher, HDR]
Guillaume Melquiond [Inria, Researcher, HDR]

Faculty Members

Sylvain Conchon [Univ Paris-Sud, Professor, HDR]
Thibault Hilaire [Délégation de l'Univ Pierre et Marie Curie, Associate Professor, until Aug 2019]
Andrei Paskevich [Univ Paris-Sud, Associate Professor]

Post-Doctoral Fellows

Cláudio Lourenço [Univ Paris-Sud, until Aug 2019]
Florian Steinberg [Inria, until Sep 2019]

PhD Students

Albin Coquereau [École Nationale Supérieure de Techniques Avancées, until Jun 2019]
Florian Faissole [Inria, until Sep 2019]
Diane Gallois-Wong [Univ Paris-Sud]
Quentin Garchery [Univ Paris-Sud]
Antoine Lanco [Inria, from Oct 2019]
Mário Pereira [Grant Portuguese government, until Apr 2019]
Raphaël Rieu-Helft [CIFRE TrustInSoft]
Mattias Roux [Univ Paris-Sud]

Technical staff

Benedikt Becker [Inria, Engineer]
Cláudio Lourenço [Inria, Engineer, from Sep 2019]
Sylvain Dailler [Inria, Engineer]
Rehan Malak [Inria, Engineer, from Apr 2019 until Sep 2019]

Administrative Assistant

Katia Evrat

External Collaborators

Thibaut Balabonski [Univ Paris-Sud, Associate Professor]
Chantal Keller [Univ Paris-Sud, Associate Professor]
Benjamin Farinier [Univ Grenoble Alpes, A.T.E.R, until Sep 2019]

2. Overall Objectives

2.1. Presentation

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 its intended behavior. Such safety-critical software appears in many application domains like transportation (e.g., aviation, aerospace, railway, and more and more in cars), communication (e.g., internet, smartphones), health devices, 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, which take a large part of the costs of software development. Yet they cannot ensure that all the bugs are caught, and remaining bugs may have catastrophic causes (e.g., the Heartbleed bug in OpenSSL library discovered in 2014 https://en.wikipedia.org/wiki/Heartbleed).

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 automatic 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 other ones in some cases.

In the past decade, there have 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 [50] and other railway-related systems; a formally proved C compiler was developed using the Coq proof assistant [63]; 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 [62]. A bug in the JDK implementation of TimSort was discovered using the KeY environment [69] and a fixed version was proved sound. Another sign of recent progress is the emergence of deductive verification competitions (e.g., VerifyThis [1]). Finally, recent trends in the industrial practice for development of critical software is to require more and more guarantees of safety, e.g., the new 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.

3. Research Program

3.1. Research Program

3.1.1. Panorama of 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 [49], JML [56]) and automatically producing a set of mathematical formulas (the *verification conditions*) which are typically proved using automatic provers (e.g., Z3 [70], Alt-Ergo [58], CVC4 [48]).

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 generally 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.

3.1.2. Overall Goals of the Toccata Project

One of our original skills is 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. Indeed, an axis of research of Toccata is the development of languages, methods and tools that are themselves formally proved correct.

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 [47], various examples of applications of that approach [54], the use of the Gappa solver for proving numerical

algorithms [68], an approach to take architectures and compilers into account when dealing with floating-point programs [55], [66]. We also contributed to the Handbook of Floating-Point Arithmetic [65]. 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 [52] [51]. Our experience led us to a conclusion that verification of numerical programs can benefit a lot from combining automatic and interactive theorem proving [53], [54], [59]. Verification of numerical programs is another main axis of Toccata.

Our scientific programme detailed below is structured into four axes:

1. Foundations and spreading of deductive program verification;
2. Reasoning on mutable memory in program verification;
3. Verification of Computer Arithmetic;
4. Spreading Formal Proofs.

Let us conclude with more general considerations about our agenda of the next four years: we want to keep on

- with general audience actions;
- industrial transfer, in particular through an extension of the perimeter of the ProofInUse joint lab.

3.2. Foundations and spreading of deductive program verification

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

This axis covers the central theme of the team: deductive verification, from the point of view of its foundations but also our will to spread its use in software development. The general motto we want to defend is “deductive verification for the masses”. A non-exhaustive list of subjects we want to address is as follows.

- The verification of general-purpose algorithms and data structures: the challenge is to discover adequate invariants to obtain a proof, in the most automatic way as possible, in the continuation of the current VOCaL project and the various case studies presented in Axis 4 below.
- Uniform approaches to obtain correct-by-construction programs and libraries, in particular by automatic extraction of executable code (in OCaml, C, CakeML, etc.) from verified programs, and including innovative general methods like advanced ghost code, ghost monitoring, etc.
- Automated reasoning dedicated to deductive verification, so as to improve proof automation; improved combination of interactive provers and fully automated ones, proof by reflection.
- Improved feedback in case of proof failures: based on generation of counterexamples, or symbolic execution, or possibly randomized techniques à la quickcheck.
- Reduction of the trusted computing base in our toolchains: production of certificates from automatic proofs, for goal transformations (like those done by Why3), and from the generation of VCs

A significant part of the work achieved in this axis is related to the Why3 toolbox and its ecosystem, displayed on Figure 1. The boxes in red background correspond to the tools we develop in the Toccata team.

3.3. Reasoning on mutable memory in program verification

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

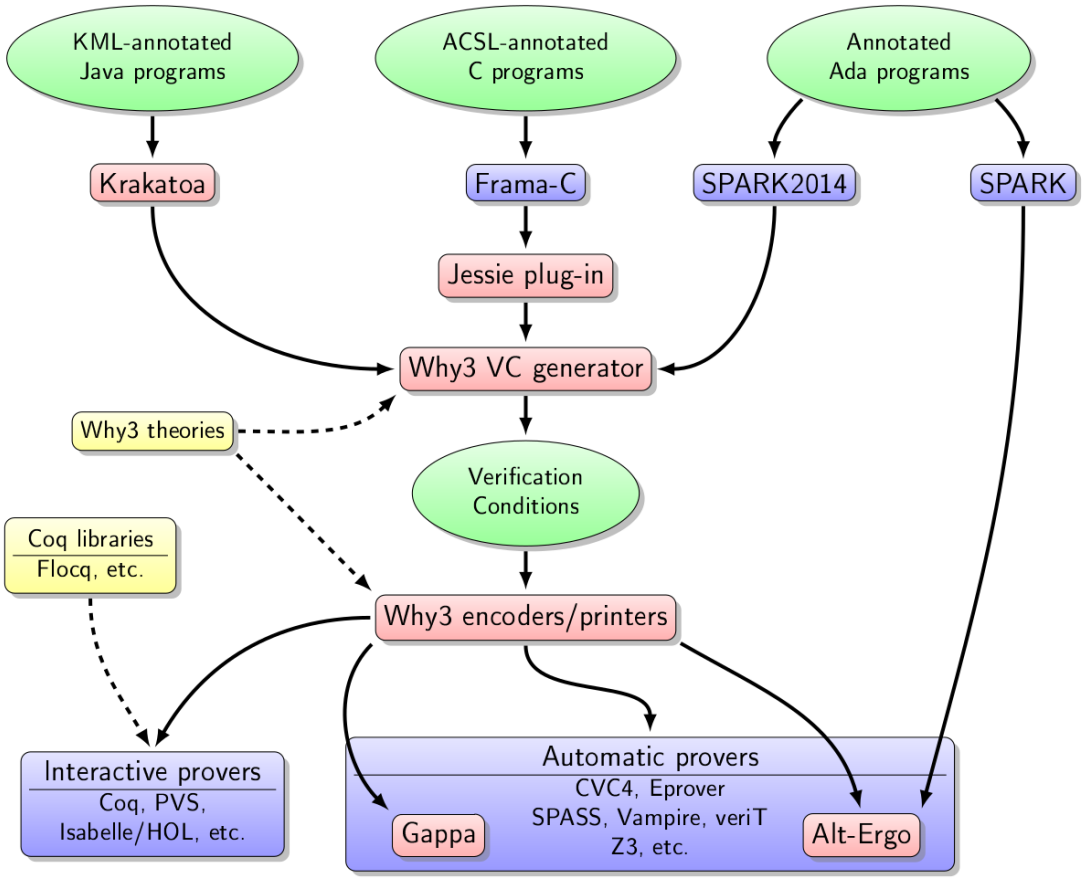

Figure 1. The Why3 ecosystem

This axis concerns specifically the techniques for reasoning on programs where aliasing is the central issue. It covers the methods based on type-based alias analysis and related memory models, on specific program logics such as separation logics, and extended model-checking. It concerns the application on analysis of C or C++ codes, on Ada codes involving pointers, but also concurrent programs in general. The main topics planned are:

- The study of advanced type systems dedicated to verification, for controlling aliasing, and their use for obtaining easier-to-prove verification conditions. Modern typing system in the style of Rust, involving ownership and borrowing, will be considered.
- The design of front-ends of Why3 for the proofs of programs where aliasing cannot be fully controlled statically, via adequate memory models, aiming in particular at extraction to C; and also for concurrent programs.
- The continuation of fruitful work on concurrent parameterized systems, and its corresponding specific SMT-based model-checking.
- Concurrent programming on weak memory models, on one hand as an extension of parameterized systems above, but also in the specific context of OCaml multicore (http://ocamllabs.io/doc/multicore.html).
- In particular in the context of the ProofInUse joint lab, design methods for Ada, C, C++ or Java using memory models involving fine-grain analysis of pointers. Rust programs could be considered as well.

3.4. Verification of Computer Arithmetic

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

We of course want to keep this axis which is a major originality of Toccata. The main topics of the next 4 years will be:

- Fundamental studies concerning formalization of floating-point computations, algorithms, and error analysis. Related to numerical integration, we will develop the relationships between mathematical stability and floating-point stability of numerical schemes.
- A significant effort dedicated to verification of numerical programs written in C, Ada, C++. This involves combining specifications in real numbers and computation in floating-point, and underlying automated reasoning techniques with floating-point numbers and real numbers. A new approach we have in mind concerns some variant of symbolic execution of both code and specifications involving real numbers.
- We have not yet studied embedded systems. Our approach is first to tackle numerical filters. This requires more results on fixed-point arithmetic and a careful study of overflows.
- Also a specific focus on arbitrary precision integer arithmetic, in the continuation of the ongoing PhD thesis of R. Rieu-Helft.

3.5. Spreading Formal Proofs

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

This axis covers applications in general. The applications we currently have in mind are:

- Hybrid Systems, i.e., systems mixing discrete and continuous transitions. This theme covers many aspects such as general techniques for formally reasoning of differential equations, and extending SMT-based reasoning. The challenge is to support both abstract mathematical reasoning and concrete program execution (e.g., using floating-point representation). Hybrid systems will be a common effort with other members of the future laboratory joint with LSV of ENS Cachan.

- Applied mathematics, in the continuation of the current efforts towards verification of Finite Element Method. It has only been studied in the mathematical point of view during this period. We plan to also consider their floating-point behavior and a demanding application is that of molecular simulation exhibited in the new EMC2 project. The challenge here is both in the mathematics to be formalized, in the numerical errors that have never been studied (and that may be huge in specific cases), and in the size of the programs, which requires that our tools scale.
- Continuation of our work on analysis of shell scripts. The challenge is to be able to analyze a large number of scripts (more than 30,000 in the corpus of Debian packages installation scripts) in an automatic manner. An approach that will be considered is some form of symbolic execution.
- Explore proof tools for mathematics, in particular automated reasoning for real analysis (application: formalization of the weak Goldbach conjecture), and in number theory.
- Obtain and distribute verified OCaml libraries, as expected outcome of the VOCaL project.
- Formalization of abstract interpretation and WP calculi: in the continuation of the former project Verasco, and an ongoing project proposal joint with CEA List. The difficulty of achieving full verification of such tools will be mitigated by use of certificate techniques.

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 or impact mainly belong to the domain of transportation: aerospace, aviation, railway, automotive.

Transfer to aeronautics: Airbus France Development of the control software of Airbus planes historically includes advanced usage of formal methods. A first aspect is the usage of the CompCert verified compiler for compiling C source code. Our work in cooperation with Gallium team for the safe compilation of floating-point arithmetic operations [2] is directly in application in this context. A second aspect is the usage of the Frama-C environment for static analysis to verify the C source code. In this context, both our tools Why3 and Alt-Ergo are indirectly used to verify C code.

Transfer to the community of Atelier B In the former ANR project BWare, we investigated 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 https://www.atelierb.eu/en/. The transfer effort continues nowadays through the FUI project LCHIP.

ProofInUse joint lab: transfer to the community of Ada development Through the creation of the ProofInUse joint lab (https://www.adacore.com/proofinuse) in 2014, with AdaCore company (https://www.adacore.com/), we have a growing impact on the community of industrial development of safety-critical applications written in Ada. See the web page https://www.adacore.com/industries for a an overview of AdaCore's customer projects, in particular those involving the use of the SPARK Pro tool set. This impact involves both the use of Why3 for generating VCs on Ada source codes, and the use of Alt-Ergo for performing proofs of those VCs.

The impact of ProofInUse can also be measured in term of job creation: the first two ProofInUse engineers, D. Hauzar and C. Fumex, employed initially on Inria temporary positions, have now been hired on permanent positions in AdaCore company. It is also interesting to notice that this effort allowed AdaCore company to get new customers, in particular the domains of application of deductive formal verification went beyond the historical domain of aerospace: application in automotive (https://www.adacore.com/customers/toyota-itc-japan) cyber-security (https://www.adacore.com/customers/multi-level-security-workstation), health (artificial heart, https://www.adacore.com/customers/total-artificial-heart).

Extension of ProofInUse joint lab The current plans for continuation of the ProofInUse joint lab (https://why3.gitlabpages.inria.fr/proofinuse/) include extension at a larger perimeter than Ada applications. We started to collaborate with the TrustInSoft company (https://trust-in-soft.com/) for the verification of C and C++ codes, including the use of Why3 to design verified and reusable C libraries (ongoing CIFRE PhD thesis). We also started to collaborate with Mitsubishi Electric in Rennes (http://www.mitsubishielectric-rce.eu/xindex.php) for a specific usage of Why3 for verifying embedded devices (logic controllers).

Generally speaking, we believe that our increasing industrial impact is a representative success for our general goal of spreading deductive verification methods to a larger audience, and we are firmly engaged into continuing such kind of actions in the future.

5. Highlights of the Year

5.1. Highlights of the Year

- Martin Clochard has been awarded the GDR GPL 2018 prize (http://gdr-gpl.cnrs.fr/node/361) for his thesis entitled "Methods and tools for specification and proof of difficult properties of sequential programs" carried out at LRI, under the scientific supervision of Claude Marché and Andrei Paskevich. [57] [43]

  Martin Clochard is currently a postdoc at ETH Zurich.

- Jean-Christophe Filliâtre receives the 2019 CAV Award, jointly with Rustan Leino (Amazon Web Services), for the design and development of reusable intermediate verification languages which significantly simplified and accelerated the building of automated deductive verifiers. Jean-Christophe is the initial designer of the Why environment for automated deductive verification, and a leading developer of its successor Why3.

  The CAV award is given anually at the CAV conference for fundamental contributions to the field of Computer-Aided Verification. http://cavconference.org/cav-award

- Claude Marché received the FIEEC CARNOT 2019 prize for applied research for his collaboration with AdaCore. The award recognizes his collaboration with AdaCore, a computer assisted proof verification company, for applications in the development of critical software for safety and security in the aeronautics, space, air traffic control and rail transportation industries, autonomous vehicles, finance or medical devices. https://www.instituts-carnot.eu/fr/actualite/prix-fieec-carnot-de-la-recherche-appliquee-trois-chercheurs-recompenses-pour-leurs-partenariats-retd-avec-les-pme

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.

RELEASE FUNCTIONAL DESCRIPTION: the "SAT solving" part can now be delegated to an external plugin, new experimental SAT solver based on mini-SAT, provided as a plugin. This solver is, in general, more efficient on ground problems, heuristics simplification in the default SAT solver and in the matching (instantiation) module, re-implementation of internal literals representation, improvement of theories combination architecture, rewriting some parts of the formulas module, bugfixes in records and numbers modules, new option "-no-Ematching" to perform matching without equality reasoning (i.e. without considering "equivalence classes"). This option is very useful for benchmarks coming from Atelier-B, two new experimental options: "-save-used-context" and "-replay-used-context". When the goal is proved valid, the first option allows to save the names of useful axioms into a ".used" file. The second one is used to replay the proof using only the axioms listed in the corresponding ".used" file. Note that the replay may fail because of the absence of necessary ground terms generated by useless axioms (that are not included in .used file) during the initial run.

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

6.2. CoqInterval

Interval package for Coq

KEYWORDS: Interval arithmetic - Coq

FUNCTIONAL DESCRIPTION: CoqInterval is a library for the proof assistant Coq.

It provides several tactics for proving 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 in Coq. In 2014, this library has been included in CoqInterval.

- Participants: Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Jean-Michel Muller, Laurence Rideau, Laurent Théry, Micaela Mayero, Mioara Joldes, Nicolas Brisebarre and Thomas Sibut-Pinote
- Contact: Guillaume Melquiond
- Publications: Proving bounds on real-valued functions with computations - Floating-point arithmetic in the Coq system - Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq - Formally Verified Approximations of Definite Integrals - Formally Verified Approximations of Definite Integrals
- URL: http://coq-interval.gforge.inria.fr/

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

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

6.5. 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 verify numerical applications.

Flocq is currently used by the CompCert verified compiler to support floating-point computations.

- Participants: Guillaume Melquiond, Pierre Roux and Sylvie Boldo
- Contact: Sylvie Boldo
- Publications: Flocq: A Unified Library for Proving Floating-point Algorithms in Coq - A Formally-Verified C Compiler Supporting Floating-Point Arithmetic - Verified Compilation of Floating-Point Computations - Innocuous Double Rounding of Basic Arithmetic Operations - Formal Proofs of Rounding Error Bounds : With application to an automatic positive definiteness check - Computer Arithmetic and Formal Proofs : Verifying Floating-point Algorithms with the Coq System
- URL: http://flocq.gforge.inria.fr/

6.6. 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 formally verifying 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 verify 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.

- Participant: Guillaume Melquiond
- Contact: Guillaume Melquiond

- Publications: Generating formally certified bounds on values and round-off errors - Formal certification of arithmetic filters for geometric predicates - Assisted verification of elementary functions - From interval arithmetic to program verification - Formally Certified Floating-Point Filters For Homogeneous Geometric Predicates - Combining Coq and Gappa for Certifying Floating-Point Programs - Handbook of Floating-Point Arithmetic - Certifying the floating-point implementation of an elementary function using Gappa - Automations for verifying floating-point algorithms - Automating the verification of floating-point algorithms - Computer Arithmetic and Formal Proofs : Verifying Floating-point Algorithms with the Coq System
- URL: http://gappa.gforge.inria.fr/

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

6.8. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes, - a critical bug fix related to template polymorphism, - native 63-bit machine integers, - a new sort of definitionally proof-irrelevant propositions: SProp, - private universes for opaque polymorphic constants, - string notations and numeral notations, - a new simplex-based proof engine for the tactics lia, nia, lra and nra, - new introduction patterns for SSReflect, - a tactic to rewrite under binders: under, - easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

7. New Results

7.1. Foundations and Spreading of Deductive Program Verification

A Why3 Framework for Reflection Proofs and its Application to GMP's Algorithms Earlier works using Why3 showed that automatically verifying the algorithms of the arbitrary-precision integer library GMP exceeds the current capabilities of automatic solvers. To complete this verification, numerous cut indications had to be supplied by the user, slowing the project to a crawl. G. Melquiond and R. Rieu-Helft extended Why3 with a framework for proofs by reflection, with minimal impact on the trusted computing base. This framework makes it easy to write dedicated decision procedures that make full use of Why3's imperative features and are formally verified. This approach opens the way to efficiently tackling the further verification of GMP's algorithms [33].

GOSPEL - Providing OCaml with a Formal Specification Language In the context of the ANR project "VOCaL" (see Sec. 9.2.2), which aims at building a formally verified OCaml library of data structures and algorithms, a specification language for OCaml is designed and implemented. It is called GOSPEL, for Generic Ocaml SPEcification Language. During his post-doc in the Toccata team, from September 2018 to August 2019, C. Lourenço implemented a parser and type checker for GOSPEL. The work on the GOSPEL language has been presented at FM'19 [23]. J.-C. Filliâtre was keynote speaker at iFM 2019 and he gave a talk on the on-going work in the VOCaL project, including GOSPEL [17].

Program Verification Competition VerifyThis 2018 VerifyThis 2018 took place on April 14 and 15, 2018 in Thessaloniki, Greece, as a satellite event of ETAPS 2018. It was the sixth edition in the VerifyThis annual competition series. Typical challenges in the VerifyThis competitions are small but intricate algorithms given in pseudo-code with an informal specification in natural language. Participants have to formalize the requirements, implement a solution, and formally verify the implementation for adherence to the specification. There are no restrictions on the programming language and verification technology used. The time frame to solve each challenge is limited to 90

minutes. Submissions are judged for correctness, completeness, and elegance. The focus includes the usability of the tools, their facilities for formalizing the properties and providing helpful output.

VerifyThis 2018 consisted of three increasingly difficult verification challenges, selected to showcase various aspects of software verification. Eleven teams (one or two participants) took part in the competition. A full report on the VerifyThis 2018 event [39] provides a presentation of the competing teams, a description of the challenges, a high-level overview of the solutions, and the results of the competition.

Proof automation with the Coq proof assistant Proof assistants based on Type Theory, such as Coq, allow the implementation of effective automatic tactics based on computational reflection. These are usually limited to a particular mathematical domain (such as linear arithmetic or ring theory). In contrast, SMTCoq is a modular and extensible tool, using external provers, which generalizes these computational approaches to combine the reasoning from multiple domains. For this, it is based on a high-level interface, which offers greater expressiveness, at the cost of more complex automation. Q. Garchery and his co-authors [22] focused on two improvements: the ability to use quantified lemmas, and the ability to use multiple representations of the same data structure. They realized a new automatic tactic, based on SMTCoq, that is expressive while keeping the modularity and the efficiency of the latter. Such a tactic thus enable scalable, low-cost automation of new domains supported by state-of-the-art automatic provers.

Certificates for Logic Transformations In a context of formal program verification, using automatic provers, the trusted code base of verification environments is typically very broad. An environment such as Why3 implements many complex procedures: generation of verification conditions, logical transformations of proof tasks, and interactions with external provers. Considering only the logical transformations of Why3, their implementation already amounts to more than 17,000 lines of OCaml code. In order to increase our confidence in the correction of such a verification tool, Garchery, Keller, Marché and Paskevich present [32] proposed a mechanism of certifying transformations, producing certificates that can be validated by an external tool, according to the *skeptical* approach. They explored two methods to validate certificates: one based on a dedicated verifier developed in OCaml, the other based on the universal proof checker Dedukti. A specificity of their certificates is to be "small grains" and composable, which makes the approach incremental, allowing to gradually add new certifying transformations.

Reasoning About Universal Cubes in MCMT The Model Checking Modulo Theories (MCMT) framework is a powerful model checking technique for verifying safety properties of parameterized transition systems. In MCMT, logical formulas are used to represent both transitions and sets of states and safety properties are verified by an SMT-based backward reachability analysis. To be fully automated, the class of formulas handled in MCMT is restricted to cubes, i.e. existentially quantified conjunction of literals. While being very expressive, cubes cannot define properties with a global termination condition, usually described by a universally quantified formula. In this work, S. Conchon and M. Roux describe BRWP, an extension of the backward reachability of MCMT for reasoning about validity properties expressed as universal cubes, that is formulas of the form $\exists i \forall j. C(i,j)$, where $C(i,j)$ is a conjunction of literals. Their approach consists in a tight cooperation between the backward reachability loop and a deductive verification engine based on weakest-precondition calculus (WP). To provide evidence for the applicability of this new algorithm, they show how to make the model checker Cubicle cooperate with Why3 [25].

A Generalized Program Verification Workflow Based on Loop Elimination and SA Form. C. Lourenço, together with Maria Frade and Jorge Sousa Pinto from Universidade do Minho, developed a minimal model of the functioning of program verification and property checking tools based on (i) the encoding of loops as non-iterating programs, either conservatively, making use of invariants and assume/assert commands, or in a bounded way; and (ii) the use of an intermediate single-assignment (SA) form. The model captures the basic workflow of tools like Boogie, Why3, or CBMC, building on a clear distinction between operational and axiomatic semantics. This allows one to consider separately the soundness of program annotation, loop encoding, translation into SA

form, and verification condition (VC) generation, as well as appropriate notions of completeness for each of these processes. To the best of our knowledge, this is the first formalization of a bounded model checking of software technique, including soundness and completeness proofs using Hoare logic; they also give the first completeness proof of a deductive verification technique based on a conservative encoding of invariant-annotated loops with assume/assert in SA form, as well as the first soundness proof based on a program logic. [21]

7.2. Reasoning on mutable memory in program verification

Certified Symbolic Execution Engine using Ghost Code Symbolic execution amounts to representing sets of concrete program states as a logical formula relating the program variables, and interpreting sets of executions as a transformation of that formula. B. Becker and C. Marché formalised the correctness of a symbolic interpreter engine, expressed by an over-approximation property stating that symbolic execution covers all concrete executions, and an under-approximation property stating that no useless symbolic states are generated. This formalisation is tailored for automated verification, that is the automated discharge of verification conditions to SMT solvers. To achieve this level of automation, they appropriately annotated the code of the symbolic interpreter with an original use of both ghost data and ghost statements [20].

Ghost Monitors M. Clochard, C. Marché and A. Paskevich proposed a new approach to deductive program verification based on auxiliary programs called *ghost monitors*. This technique is useful when the syntactic structure of the target program is not well suited for verification, for example, when an essentially recursive algorithm is implemented in an iterative fashion. This new approach consists in implementing, specifying, and verifying an auxiliary program that monitors the execution of the target program, in such a way that the correctness of the monitor entails the correctness of the target. The ghost monitor maintains the necessary data and invariants to facilitate the proof. It can be implemented and verified in any suitable framework, which does not have to be related to the language of the target programs. This technique is also applicable when one wants to establish relational properties between two target programs written in different languages and having different syntactic structure.

Ghost monitors can be used to specify and prove fine-grained properties about the *infinite behaviors* of target programs. Since this cannot be easily done using existing verification frameworks, this work introduces a dedicated language for ghost monitors, with an original construction to *catch* and handle divergent executions. The soundness of the underlying program logic is established using a particular flavor of transfinite games. This language and its soundness are formalized and mechanically checked. [24]

7.3. Verification of Computer Arithmetic

Formal Verification of a State-of-the-Art Integer Square Root Even though some programs only use integer operations, the best way to understand and verify them might be to view them as fixed-point arithmetic algorithm. This is the case of the function from the GMP library that computes the square root of a 64-bit integer. The C code is short but intricate, as it implements Newton's method and it relies on magic constants and intentional arithmetic overflows. G. Melquiond and R. Rieu-Helft have verified this algorithm using the Why3 tool and automated solvers such as Gappa [28].

Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods S. Boldo, F. Faissole, and A. Chapoutot developed a new fine-grained analysis of round-off errors in explicit Runge-Kutta integration methods, taking into account exceptional behaviors, such as underflow and overflow [12]. First steps towards the formalization has been done by F. Faissole [34].

Optimal Inverse Projection of Floating-Point Addition In a setting where we have intervals for the values of floating-point variables x, a, and b, we are interested in improving these intervals when the floating-point equality $x \oplus a = b$ holds. This problem is common in constraint propagation, and called the inverse projection of the addition. D. Gallois-Wong, S. Boldo, and P. Cuoq proposed floating-point theorems that provide optimal bounds for all the intervals [13].

Emulating round-to-nearest-ties-to-zero "augmented" floating- point operations using round-to-nearest-ties-to-even arit The 2019 version of the IEEE 754 Standard for Floating-Point Arithmetic recommends that new "augmented" operations should be provided for the binary formats. These operations use a new "rounding direction": round to nearest *ties-to-zero*. S. Boldo, C. Lauter, and J.-M. Muller show how they can be implemented using the currently available operations, using round-to-nearest *ties-to-even* with a partial formal proof of correctness [42].

LTI filters Several developments were made towards the efficency and accuracy of the implementation of LTI (linear time-invariant) numerical filters: a word-length optimization problem under accuracy constraints [26] by T. Hilaire, H. Ouzia, and B. Lopez, and a tight worst-case error analysis [16] by A. Volkova, T. Hilaire, and C. Lauter.

7.4. Spreading Formal Proofs

7.4.1. Real Analysis

Formally Verified Approximations of Definite Integrals The CoqInterval library provides some tactics for computing and formally verifying numerical approximations of real-valued expressions inside the Coq system. In particular, it is able to compute reliable bounds on proper definite integrals [64]. A. Mahboubi, G. Melquiond, and T. Sibut-Pinote extended these algorithms to also cover some improper integrals, e.g., those with an unbounded integration domain [14]. This makes CoqInterval one of the very few tools able to produce reliable results for improper integrals, be they formally verified or not.

Coq Formalization of algorithms for numerical filters D. Gallois-Wong developed a Coq formalization of a generic representation of numerical filters, called SIF [31] in order to encompass all other representations of filters, and prove useful theorems only once.

Complexity theory and constructive analysis E. Neumann and F. Steinberg extended the framework for complexity of operators in analysis devised by Kawamura and Cook (2012) to allow for the treatment of a wider class of representations and applied it to the study of interval computation [15]. A. Kawamura, F. Steinberg, and H. Thies put forward a complexity class of type-two linear-time [27].

F. Steinberg, L. Théry, and H. Thies give a number of formal proofs of theorems from the field of computable analysis. Results include that the algebraic operations and the efficient limit operator on the reals are computable, that certain countably infinite products are isomorphic to spaces of functions, compatibility of the enumeration representation of subsets of natural numbers with the abstract definition of the space of open subsets of the natural numbers, and that continuous realizability implies sequential continuity [46] [29]. F. Steinberg and H. Thies formalized proofs about Baire spaces and the isomorphy of the concrete and abstract spaces of open sets [45].

7.4.2. Formal Analysis of Debian packages

Several new results were produced in the context of the CoLiS project for the formal analysis of Debian packages. A first important step is the version 2 of the design of the CoLiS language done by B. Becker, C. Marché and other co-authors [38], that includes a modified formal syntax, a extended formal semantics, together with the design of concrete and symbolic interpreters. Those interpreters are specified and implemented in Why3, proved correct (following the initial approach for the concrete interpreter published in 2018 [60] and the recent approach for symbolic interpretation mentioned above [20]), and finally extracted to OCaml code.

To make the extracted code effective, it must be linked together with a library that implements a solver for feature constraints [61], and also a library that formally specifies the behavior of basic UNIX utilities. The latter library is documented in details in a research report [40].

A third result is a large verification campaign running the CoLiS toolbox on all the packages of the current Debian distribution. The results of this campaign were reported in another article [41] that will be presented at TACAS conference in 2020. The most visible side effect of this experiment is the discovery of bugs: more than 150 bugs report have been filled against various Debian packages.

7.4.3. Miscellaneous

Functional Programming. J.-C. Filliâtre was invited speaker at JFLA 2019, as part of a session celebrating the 30 years of JFLA (a French-speaking national conference related to functional programming). He talked about 25 years of programming with OCaml [18]. At JFLA 2020, J.-C. Filliâtre will give a talk related to the elimination of non-tail calls [30].

Formal Verification of "ParcourSup" algorithms. In May–July 2019, Léo Andrès (M1 student at Paris Sud) did a three month internship on the verification of the first algorithm of Parcoursup using Why3. Most of the expected properties, taken from the public description of Parcoursup's algorithms, have been verified. Léo Andrès's report (in French), is available on-line [37]. In June-December 2019, Benedikt Becker worked on the verification of the Java source code of ParcourSup. The findings and lessons learnt are described in a report under preparation.

Formalizing loop-carried dependencies in Coq for high-level synthesis. F. Faissole, G. Constantinides, and D. Thomas developed Coq formalizations in order to improve high-level synthesis for FPGAs [44].

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

We have two bilateral contracts which are closely related to a joint effort called the ProofInUse joint Laboratory. The objective of ProofInUse is to provide verification tools, based on mathematical proof, to industry users. These tools are aimed at replacing or complementing the existing test activities, whilst reducing costs.

This joint laboratory is a follow-up of the former "LabCom ProofInUse" between Toccata and the SME AdaCore, funded by the ANR programme "Laboratoires communs", from April 2014 to March 2017 http://www.spark-2014.org/proofinuse.

8.1.1. ProofInUse-AdaCore Collaboration

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

This collaboration is a joint effort of the Inria project-team Toccata and the AdaCore company which provides development tools for the Ada programming language. It is funded by a 5-year bilateral contract from Jan 2019 to Dec 2023.

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 objective of ProofInUse-AdaCore is to significantly increase the capabilities and performances of the Spark/Ada verification environment proposed by AdaCore. It aims at integration of verification techniques at the state-of-the-art of academic research, via the generic environment Why3 for deductive program verification developed by Toccata.

8.1.2. ProofInUse-MERCE Collaboration

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

This bilateral contract is part of the ProofInUse effort. This collaboration joins efforts of the Inria project-team Toccata and the company Mitsubishi Electric R&D (MERCE) in Rennes. It is funded by a bilateral contract of 18 months from Nov 2019 to April 2021.

MERCE has strong and recognized skills in the field of formal methods. In the industrial context of the Mitsubishi Electric Group, MERCE has acquired knowledge of the specific needs of the development processes and meets the needs of the group in different areas of application by providing automatic verification and demonstration tools adapted to the problems encountered.

The objective of ProofInUse-MERCE is to significantly improve on-going MERCE tools regarding the verification of Programmable Logic Controllers and also regarding the verification of numerical C codes.

8.2. Bilateral Grants with Industry

8.2.1. CIFRE contract with TrustInSoft company

Participants: Guillaume Melquiond [contact], Raphaël Rieu-Helft.

Jointly with the thesis of R. Rieu-Helft, supervised in collaboration with the TrustInSoft company, we established a 3-year bilateral collaboration contract, that started in October 2017. The aim is to design methods that make it possible to design an arbitrary-precision integer library that, while competitive with the state-of-the-art library GMP, is formally verified. Not only are GMP's algorithm especially intricate from an arithmetic point of view, but numerous tricks were also used to optimize them. We are using the Why3 programming language to implement the algorithms, we are developing reflection-based procedures to verify them, and we finally extract them as a C library that is binary-compatible with GMP [9] [67] [33].

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. 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)

9.1.2. MILC

Participant: Sylvie Boldo [contact].

MILC is a DIM-RFSI project. It is a one-year project (2018–2019) that aims at formalizing measure theory and Lebesgue integral in the Coq proof assistant. https://lipn.univ-paris13.fr/MILC/

Partners: Université Paris 13 (M. Mayero, PI), Inria Paris, Inria Saclay

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 60 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 60 months, starting on October 1st, 2015. See 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. 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.

9.2.4. ANR PARDI

Participant: Sylvain Conchon [contact].

Verification of PARameterized DIstributed systems. A parameterized system specification is a specification for a whole class of systems, parameterized by the number of entities and the properties of the interaction, such as the communication model (synchronous/asynchronous, order of delivery of message, application ordering) or the fault model (crash failure, message loss). To assist and automate verification without parameter instantiation, PARDI uses two complementary approaches. First, a fully automatic model checker modulo theories is considered. Then, to go beyond the intrinsic limits of parameterized model checking, the project advocates a collaborative approach between proof assistant and model checker. http://pardi.enseeiht.fr/

The proof lead by Toulouse INP/IRIT started in 2016 and lasts for 4 years. Partners: Université Pierre et Marie Curie (LIP6), Université Paris-Sud (LRI), Inria Nancy (team VERIDIS)

9.3. European Initiatives

9.3.1. FP7 & H2020 Projects

9.3.1.1. EMC2

Participant: Sylvie Boldo [contact].

A new ERC Synergy Grant 2018 project, called Extreme-scale Mathematically-based Computational Chemistry (EMC2) has just been accepted. The PIs are É. Cancès, L. Grigori, Y. Maday and J.-P. Piquemal. S. Boldo is part of the work package 3: validation and certification of molecular simulation results. https://www.sorbonne-universite.fr/newsroom/actualites/erc-synergy-grant-2018

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.

9.4. International Research Visitors

9.4.1. Visits of International Scientists

Jorge Sousa Pinto, professor from Universidade do Minho (Braga, Portugal, https://haslab.uminho.pt/jsp/) visited the team for 1 month in May 2019. We interact with him on the topic of the formalization of VC generation algorithms [21]. He also proposed a formalization using the Why3 tool.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. General Chair, Scientific Chair

J.-C. Filliâtre, scientific chair and co-organizer of EJCP (École Jeunes Chercheurs en Programmation du GDR GPL) at Strasbourg on June 24–28, 2019. 5 days / 8 lectures / 25 participants. http://ejcp2019.icube.unistra.fr/

10.1.2. Scientific Events: Selection

10.1.2.1. Chair of Conference Program Committees

S. Boldo, program co-chair of the 26th IEEE Symposium on Computer Arithmetic (ARITH 2019), Kyoto, Japan. [35]

10.1.2.2. Member of the Conference Program Committees

S. Boldo, 11th NASA Formal Methods Symposium (NFM 2019)

J.-C. Filliâtre, 10th International Conference on Interactive Theorem Proving (ITP 2019)

J.-C. Filliâtre, 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)

J.-C. Filliâtre, European Symposium on Programming (ESOP 2020)

G. Melquiond, 26th IEEE Symposium on Computer Arithmetic (ARITH 2019)

G. Melquiond, 10th International Conference on Interactive Theorem Proving (ITP 2019)

10.1.2.3. Reviewer

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

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

G. Melquiond, member of the editorial board of *Reliable Computing*.

J.-C. Filliâtre, member of the editorial board of *Journal of Functional Programming*.

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

G. Melquiond, “Computer Arithmetic and Formal Proofs” [19], at the 30th Journées Francophones des Langages Applicatifs (JFLA 2019).

J.-C. Filliâtre, “Deductive Verification of OCaml Libraries” [17], at the 15th International Conference on integrated Formal Methods (iFM), December 2019, Bergen, Norway.

J.-C. Filliâtre, “Retour sur 25 ans de programmation avec OCaml” [18], at Journées Francophones des Langages Applicatifs (JFLA), January 2019, Les Rousses, France.

J.-C. Filliâtre, “The Why3 tool for deductive verification and verified OCaml libraries”, at Frama-C/Spark day 2019, Paris, France.

10.1.5. Leadership within the Scientific Community

S. Boldo, elected chair of the ARITH working group of the GDR-IM (a CNRS subgroup of computer science) with J. Detrey (Inria Nancy) and now L.-S. Didier (Univ. Toulon).

J.-C. Filliâtre, chair of IFIP WG 1.9/2.15 verified Software.

10.1.6. Scientific Expertise

G. Melquiond, member of the scientific commission of Inria-Saclay, in charge of selecting candidates for PhD grants, Post-doc grants, temporary leaves from universities (“délégations”).

S. Boldo, member of the program committee for selecting postdocs of the maths/computer science program of the Labex mathématique Hadamard.

S. Boldo has mentored for 1 year a female PhD student of University Paris-Saclay (from epidemiology).

J.-C. Filliâtre, grading the entrance examination at X/ENS (*“option informatique”*).

C. Marché, member of DigiCosme committee for research and innovation (selection of projects for working groups, post-doc grants, doctoral missions, invited professors)

10.1.7. Research Administration

G. Melquiond, member of the committee for the monitoring of PhD students (*"commission de suivi doctoral"*).

S Boldo, member of the CLFP (*"commission locale de formation permanente"*).

S. Boldo, member of the (national) IES commission.

S. Boldo, member of the CDT commission of Saclay (*"commission de développement technologique"*).

S. Boldo, member of the STIC department commission of Univ. Paris-Sud.

S. Boldo, member of the executive commission for the Digicosme Labex.

S. Boldo, deputy scientific director (DSA) of Inria Saclay research center from January 1st, 2019

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. Lanco, *Unix et programmation Web*, 24h, L2, Université Paris-Saclay, France.

G. Melquiond, *Programmation C++ avancée*, 12h, M2, Université Paris-Saclay, France.

J.-C. Filliâtre, *Langages de programmation et compilation*, 25h, L3, École Normale Supérieure, France.

J.-C. Filliâtre, *Les bases de l'algorithmique et de la programmation*, 15h, L3, École Polytechnique, France.

J.-C. Filliâtre, *Compilation*, 18h, M1, École Polytechnique, France.

S. Boldo was invited in the Xavier Leroy's course at Collège de France on Dec 19th.

10.2.2. Supervision

HDR: G. Melquiond, "Formal Verification for Numerical Computations, and the Other Way Around" [11], Université Paris-Sud, Apr. 1st, 2019.

PhD: F. Faissole, "Formalisations d'analyses d'erreurs en analyse numérique et en arithmétique à virgule flottante", Université Paris-Saclay & Université Paris-Sud, Dec 13th 2019, supervised by S. Boldo and A. Chapoutot. [Not yet on HAL]

PhD: A. Coquereau, "Amélioration de performances du solveur SMT Alt-Ergo grâce à l'intégration d'un solveur SAT efficace", Université Paris-Saclay & Université Paris-Sud, Dec 16th 2019, supervised by S. Conchon, F. Le Fessant et M. Mauny. [Not yet on HAL]

PhD: M. Roux, "Extensions de l'algorithme d'atteignabilité arrière dans le cadre de la vérification de modèles modulo théories", Université Paris-Saclay & Université Paris-Sud, Dec 19th 2019, supervised by Sylvain Conchon. [Not yet on HAL]

PhD in progress: R. Rieu-Helft, "Développement et vérification de bibliothèques d'arithmétique entière en précision arbitraire", since Oct. 2017, supervised by G. Melquiond and P. Cuoq (TrustInSoft).

PhD in progress: D. Gallois-Wong, "Vérification formelle et filtres numériques", since Oct. 2017, supervised by S. Boldo and T. Hilaire.

PhD in progress: Q. Garchery, "Certification de la génération et de la transformation d'obligations de preuve", since Oct. 2018, supervised by C. Keller, C. Marché and A. Paskevich.

PhD in progress: A. Lanco, "Stratégies pour la réduction forte", since Oct. 2019, supervised by T. Balabonski and G. Melquiond.

G. Turbiau has been a M1 trainee for 2 months, supervised by S. Boldo.

10.2.3. Juries

S. Boldo, member of the Inria CRCN recruiting juries, national one and at Rennes.

S. Boldo, president of the PhD defense of Clothilde Jeangoudoux, Sorbonne University, May 21st

S. Boldo, reviewer of the habilitation of Christoph Lauter, Sorbonne University, May 22nd

S. Boldo, reviewer of the PhD defence of Florent Bréhard, ENS Lyon, July 12th

S. Boldo, reviewer of PhD defence of Damien Rouhling, Université Côte d'Azur, Sept 30th

S. Boldo, president of the PhD defense of Yohan Chatelain, Université Paris-Saclay, Dec 12th

S. Boldo, president of the PhD defense of Armaël Guéneau, Université de Paris, Dec 16th

S. Boldo, member of the PhD defense of Antoine Kaszczyc, Université Paris-Nord, Dec 18th

J.-C. Filliâtre, reviewer of the PhD defense of Y. El Bakouny, "Scallina: On the Intersection of Scala and Gallina", Université Saint-Joseph, Beirut, Lebanon, December 19, 2019.

J.-C. Filliâtre, president of the PhD defense of L. Blatter "Relational properties for specification and verification of C programs in Frama-C", University Paris Saclay, September 26, 2019.

10.3. Popularization

10.3.1. Internal or external Inria responsibilities

S. Boldo has been the scientific head for Saclay for the MECSI group for networking about computer science popularization inside Inria until October 2019.

She was also responsible (with A. Couvreur of the Grace team a,d M. Quet of the SCM) for the 2019 "Fête de la science" on October 10th and 11th 2019. Teenagers were welcomed on 8 activities ranging from unplugged activities with Duplo construction toys to programming, and from applied mathematics to theoretical computer science.

10.3.2. Education

S. Conchon and J.-C. Filliâtre, together with K. Nguyen and T. Balabonski (members of the LRI team but not members of Toccata), wrote a book "Numérique et Sciences Informatiques, 30 leçons avec exercices corrigés" (Ellipses, August 2019) [36] targeting the new Computer Science programme in the French high school system, which started in September 2019.

S. Boldo gave a talk and was involved in the organization of the "Rencontres des Jeunes mathématiciennes et Informaticiennes" organized in Saclay on Oct 21st and 22nd 2019. About 20 female teenagers were welcomed for two days with talks and exercises to discover research.

10.3.3. Interventions

C. Marché presented an overview of the Why3 tool and its ecosystem at the Paris Open Source Summit in Saint-Denis, Dec. 11, 2019, as part of a workshop "ATELIER Inria : OCaml, Coq, Why 3: pour concevoir codes et infrastructures plus sûrs en C, Java, Javascript, Ada" https://www.opensourcesummit.paris/conferences.html?date=1576022400).

S. Boldo gave a talk at a high school in Sceaux on Feb 8th.

S. Boldo presented the floating-point arithmetic at IRIF to the members of the CODYS ANR on April 1st.

11. Bibliography

Major publications by the team in recent years

[1] 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

[2] 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

[3] S. BOLDO, G. MELQUIOND. *Computer Arithmetic and Formal Proofs: Verifying Floating-point Algorithms with the Coq System*, ISTE Press - Elsevier, December 2017, https://hal.inria.fr/hal-01632617

[4] M. CLOCHARD, L. GONDELMAN, M. PEREIRA. *The Matrix Reproved*, in "Journal of Automated Reasoning", 2018, vol. 60, n° 3, pp. 365–383, https://hal.inria.fr/hal-01617437

[5] S. CONCHON, M. IGUERNLALA, K. JI, G. MELQUIOND, C. FUMEX. *A Three-tier Strategy for Reasoning about Floating-Point Numbers in SMT*, in "Computer Aided Verification", 2017, https://hal.inria.fr/hal-01522770

[6] J.-C. FILLIÂTRE, L. GONDELMAN, A. PASKEVICH. *The Spirit of Ghost Code*, in "Formal Methods in System Design", 2016, vol. 48, n° 3, pp. 152–174, https://hal.archives-ouvertes.fr/hal-01396864v1

[7] C. FUMEX, C. DROSS, J. GERLACH, C. MARCHÉ. *Specification and Proof of High-Level Functional Properties of Bit-Level Programs*, in "8th NASA Formal Methods Symposium", Minneapolis, MN, USA, S. RAYADURGAM, O. TKACHUK (editors), Lecture Notes in Computer Science, Springer, June 2016, vol. 9690, pp. 291–306, https://hal.inria.fr/hal-01314876

[8] É. MARTIN-DOREL, G. MELQUIOND. *Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq*, in "Journal of Automated Reasoning", 2016, https://hal.inria.fr/hal-01086460

[9] G. MELQUIOND, R. RIEU-HELFT. *A Why3 Framework for Reflection Proofs and its Application to GMP's Algorithms*, in "9th International Joint Conference on Automated Reasoning", Oxford, United Kingdom, D. GALMICHE, S. SCHULZ, R. SEBASTIANI (editors), Lecture Notes in Computer Science, July 2018, https://hal.inria.fr/hal-01699754

[10] P. ROUX, M. IGUERNLALA, S. CONCHON. *A Non-linear Arithmetic Procedure for Control-Command Software Verification*, in "24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)", Thessalonique, Greece, April 2018, https://hal.archives-ouvertes.fr/hal-01737737

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] G. MELQUIOND. *Formal Verification for Numerical Computations, and the Other Way Around*, Université Paris Sud, April 2019, Habilitation à diriger des recherches, https://tel.archives-ouvertes.fr/tel-02194683

Articles in International Peer-Reviewed Journals

[12] S. BOLDO, F. FAISSOLE, A. CHAPOUTOT. *Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods*, in "IEEE Transactions on Computers", 2019, forthcoming [*DOI :* 10.1109/TC.2019.2917902], https://hal.archives-ouvertes.fr/hal-01883843

[13] D. GALLOIS-WONG, S. BOLDO, P. CUOQ. *Optimal Inverse Projection of Floating-Point Addition*, in "Numerical Algorithms", 2019, forthcoming [*DOI :* 10.1007/s11075-019-00711-z], https://hal.inria.fr/hal-01939097

[14] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, in "Journal of Automated Reasoning", February 2019, vol. 62, n° 2, pp. 281-300 [*DOI :* 10.1007/s10817-018-9463-7], https://hal.inria.fr/hal-01630143

[15] E. NEUMANN, F. STEINBERG. *Parametrised second-order complexity theory with applications to the study of interval computation*, in "Theoretical Computer Science", 2019, forthcoming [*DOI :* 10.1016/j.tcs.2019.05.009], https://hal.inria.fr/hal-02148494

[16] A. VOLKOVA, T. HILAIRE, C. LAUTER. *Arithmetic approaches for rigorous design of reliable Fixed-Point LTI filters*, in "IEEE Transactions on Computers", 2019, pp. 1-14, forthcoming [*DOI :* 10.1109/TC.2019.2950658], https://hal.archives-ouvertes.fr/hal-01918650

Invited Conferences

[17] J.-C. FILLIÂTRE. *Deductive Verification of OCaml Libraries*, in "iFM 2019 - International Conference on integrated Formal Methods", Bergen, Norway, December 2019, https://hal.inria.fr/hal-02406253

[18] J.-C. FILLIÂTRE. *Retour sur 25 ans de programmation avec OCaml*, in "JFLA 2019 - Journées Francophones des Langages Applicatifs", Les Rousses, France, January 2019, https://hal.inria.fr/hal-02406208

[19] G. MELQUIOND. *Computer Arithmetic and Formal Proofs*, in "30èmes Journées Francophones des Langages Applicatifs", Les Rousses, France, N. MAGAUD (editor), January 2019, https://hal.inria.fr/hal-02013540

International Conferences with Proceedings

[20] B. BECKER, C. MARCHÉ. *Ghost Code in Action: Automated Verification of a Symbolic Interpreter*, in "VSTTE 2019 - 11th Working Conference on Verified Software: Tools, Techniques and Experiments", New York, United States, S. CHAKRABORTY, J. A.NAVAS (editors), Lecture Notes in Computer Science, September 2019, https://hal.inria.fr/hal-02276257

[21] C. BELO LOURENÇO, M. J. FRADE, J. SOUSA PINTO. *A Generalized Program Verification Workflow Based on Loop Elimination and SA Form*, in "FormaliSE 2019 - 7th International Conference on Formal Methods in Software Engineering", Montreal, Canada, May 2019, https://hal.inria.fr/hal-02431769

[22] V. BLOT, A. BOUSALEM, Q. GARCHERY, C. KELLER. *SMTCoq: automatisation expressive et extensible dans Coq*, in "Journées Francophones des Langages Applicatifs", Les Rousses, France, January 2019, https://hal.archives-ouvertes.fr/hal-02369249

[23] A. CHARGUÉRAUD, J.-C. FILLIÂTRE, C. LOURENÇO, M. PEREIRA. *GOSPEL -Providing OCaml with a Formal Specification Language*, in "FM 2019 - 23rd International Symposium on Formal Methods", Porto, Portugal, October 2019, https://hal.inria.fr/hal-02157484

[24] M. CLOCHARD, C. MARCHÉ, A. PASKEVICH. *Deductive Verification with Ghost Monitors*, in "Principles of Programming Languages", New Orleans, United States, 2020 [*DOI :* 10.1145/3371070], https://hal.inria.fr/hal-02368284

[25] S. CONCHON, M. ROUX. *Reasoning about Universal Cubes in MCMT*, in "Formal Methods and Software Engineering", Shenzhen, China, Lecture Notes in Computer Science, 2019, n^{o} 11852, pp. 270–285, https://hal.archives-ouvertes.fr/hal-02420588

[26] T. HILAIRE, H. OUZIA, B. LOPEZ. *Optimal Word-Length Allocation for the Fixed-Point Implementation of Linear Filters and Controllers*, in "2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)", Kyoto, Japan, IEEE, June 2019, pp. 175-182 [*DOI :* 10.1109/ARITH.2019.00040], https://hal.sorbonne-universite.fr/hal-02393851

[27] A. KAWAMURA, F. STEINBERG, H. THIES. *Second-Order Linear-Time Computability with Applications to Computable Analysis*, in "TAMC 2019 - 15th Annual Conference Theory and Applications of Models of Computation", Tokyo, Japan, T. V. GOPAL, J. WATADA (editors), LNCS - Lecture Notes in Computer Science, Springer, March 2019, vol. 11436, pp. 337-358, https://hal.inria.fr/hal-02148490

[28] G. MELQUIOND, R. RIEU-HELFT. *Formal Verification of a State-of-the-Art Integer Square Root*, in "ARITH-26 2019 - 26th IEEE 26th Symposium on Computer Arithmetic", Kyoto, Japan, S. BOLDO, M. LANGHAMMER (editors), June 2019, pp. 183-186 [*DOI :* 10.1109/ARITH.2019.00041], https://hal.inria.fr/hal-02092970

[29] F. STEINBERG, H. THIES, L. THÉRY. *Quantitative continuity and Computable Analysis in Coq*, in "ITP 2019 - Tenth International Conference on Interactive Theorem Proving", Portland, United States, 2019, The version accepted to the conference can be accessed at https://drops.dagstuhl.de/opus/volltexte/2019/11083/ [*DOI :* 10.4230/LIPICS.ITP.2019.28], https://hal.archives-ouvertes.fr/hal-02426470

National Conferences with Proceedings

[30] J.-C. FILLIÂTRE. *Mesurer la hauteur d'un arbre*, in "Journées Francophones des Langages Applicatifs", Gruissan, France, January 2020, https://hal.inria.fr/hal-02315541

[31] D. GALLOIS-WONG. *Formalisation en Coq d'algorithmes de filtres numériques*, in "Journées Francophones des Langages Applicatifs (JFLA) 2019", Les Rousses, France, Journées Francophones des Langages Applicatifs 2019, Nicolas Magaud, January 2019, https://hal.inria.fr/hal-01929531

[32] Q. GARCHERY, C. KELLER, C. MARCHÉ, A. PASKEVICH. *Des transformations logiques passent leur certicat*, in "Journées Francophones des Langages Applicatifs", Gruissan, France, 2020, https://hal.inria.fr/hal-02384946

[33] R. RIEU-HELFT. *Un mécanisme de preuve par réflexion pour Why3 et son application aux algorithmes de GMP*, in "JFLA 2019 - 30èmes Journées Francophones des Langages Applicatifs", Rousses, France, January 2019, https://hal.inria.fr/hal-01943010

Conferences without Proceedings

[34] F. FAISSOLE. *Formalisation en Coq des erreurs d'arrondi de méthodes de Runge-Kutta pour les systèmes matriciels*, in "AFADL 2019 - 18e journées Approches Formelles dans l'Assistance au Développement de Logiciels", Toulouse, France, June 2019, https://hal.archives-ouvertes.fr/hal-02391924

Scientific Books (or Scientific Book chapters)

[35] *Proceedings of the 26th Symposium on Computer Arithmetic (ARITH 2019)*, IEEE, Kyoto, June 2019 [*DOI :* 10.1109/ARITH.2019.00005], https://hal.inria.fr/hal-02433652

[36] T. BALABONSKI, S. CONCHON, J.-C. FILLIÂTRE, K. NGUYỄN. *Numérique et Sciences Informatiques, 30 leçons avec exercices corrigés*, Ellipses, August 2019, https://hal.inria.fr/hal-02379073

Research Reports

[37] L. ANDRÈS. *Vérification par preuve formelle de propriétés fonctionnelles d'algorithme de classification*, Université Paris Sud (Paris 11) - Université Paris Saclay, August 2019, https://hal.inria.fr/hal-02421484

[38] B. BECKER, N. JEANNEROD, C. MARCHÉ, R. TREINEN. *Revision 2 of CoLiS language: formal syntax, semantics, concrete and symbolic interpreters*, ANR, October 2019, https://hal.inria.fr/hal-02321743

[39] M. HUISMAN, R. MONAHAN, P. MÜLLER, A. PASKEVICH, G. ERNST. *VerifyThis 2018: A Program Verification Competition*, Université Paris-Saclay, January 2019, https://hal.inria.fr/hal-01981937

[40] N. JEANNEROD, C. MARCHÉ, Y. RÉGIS-GIANAS, M. SIGHIREANU, R. TREINEN. *Specification of UNIX Utilities*, ANR, October 2019, https://hal.inria.fr/hal-02321691

Other Publications

[41] B. BECKER, N. JEANNEROD, C. MARCHÉ, Y. RÉGIS-GIANAS, M. SIGHIREANU, R. TREINEN. *Analysing installation scenarios of Debian packages*, 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02355602

[42] S. BOLDO, C. Q. LAUTER, J.-M. MULLER. *Emulating round-to-nearest-ties-to-zero "augmented" floating-point operations using round-to-nearest-ties-to-even arithmetic*, October 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02137968

[43] M. CLOCHARD. *Méthodes et outils pour la spécification et la preuve de propriétés difficiles de programmes séquentiels (Documents de soutenance)*, February 2019, https://hal.inria.fr/hal-02047458

[44] F. FAISSOLE, G. CONSTANTINIDES, D. THOMAS. *Formalizing loop-carried dependencies in Coq for high-level synthesis*, April 2019, FCCM 2019 - 27th IEEE Symposium on Field-Programmable Custom Computing Machines, Poster, https://hal.archives-ouvertes.fr/hal-02391954

[45] F. STEINBERG, H. THIES. *Some formal proofs of isomorphy and discontinuity*, March 2019, MLA 2019 - Third Workshop on Mathematical Logic and its Applications, https://hal.inria.fr/hal-02019174

[46] F. STEINBERG, L. THÉRY, H. THIES. *Quantitative continuity and computable analysis in Coq*, April 2019, working paper or preprint, https://hal.inria.fr/hal-02088293

References in notes

[47] 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

[48] C. BARRETT, C. L. CONWAY, M. DETERS, L. HADAREAN, D. JOVANOVIĆ, T. KING, A. REYNOLDS, C. TINELLI. *CVC4*, in "Proceedings of the 23rd international conference on Computer aided verification", Berlin, Heidelberg, CAV'11, Springer-Verlag, 2011, pp. 171–177

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

[50] 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

[51] 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/

[52] 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/

[53] 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

[54] 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

[55] 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

[56] 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

[57] M. CLOCHARD. *Méthodes et outils pour la spécification et la preuve de propriétés difficiles de programmes séquentiels*, Université Paris-Saclay, March 2018, https://tel.archives-ouvertes.fr/tel-01787689

[58] S. CONCHON, A. COQUEREAU, M. IGUERNLALA, A. MEBSOUT. *Alt-Ergo 2.2*, in "SMT Workshop: International Workshop on Satisfiability Modulo Theories", Oxford, United Kingdom, July 2018, https://hal.inria.fr/hal-01960203

[59] C. FUMEX, C. MARCHÉ, Y. MOY. *Automating the Verification of Floating-Point Programs*, in "Verified Software: Theories, Tools, and Experiments. Revised Selected Papers Presented at the 9th International Conference VSTTE", Heidelberg, Germany, A. PASKEVICH, T. WIES (editors), Lecture Notes in Computer Science, Springer, December 2017, n° 10712, https://hal.inria.fr/hal-01534533/

[60] N. JEANNEROD, C. MARCHÉ, R. TREINEN. *A Formally Verified Interpreter for a Shell-like Programming Language*, in "Verified Software: Theories, Tools, and Experiments. Revised Selected Papers Presented at the 9th International Conference VSTTE", Heidelberg, Germany, A. PASKEVICH, T. WIES (editors), Lecture Notes in Computer Science, Springer, December 2017, n° 10712, https://hal.inria.fr/hal-01534747v1

[61] N. JEANNEROD, R. TREINEN. *Deciding the First-Order Theory of an Algebra of Feature Trees with Updates*, in "9th International Joint Conference on Automated Reasoning", Oxford, United Kingdom, July 2018, https://hal.archives-ouvertes.fr/hal-01807474

[62] 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° 6, pp. 107–115

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

[64] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, in "Proceedings of the 7th International Conference on Interactive Theorem Proving", Nancy, France, J. C. BLANCHETTE, S. MERZ (editors), Lecture Notes in Computer Science, August 2016, vol. 9807, https://hal.inria.fr/hal-01289616

[65] J.-M. MULLER, N. BRUNIE, F. DE DINECHIN, C.-P. JEANNEROD, M. JOLDES, V. LEFÈVRE, G. MELQUIOND, N. REVOL, S. TORRES. *Handbook of Floating-point Arithmetic (2nd edition)*, Birkhäuser Basel, July 2018, https://hal.inria.fr/hal-01766584

[66] 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

[67] R. RIEU-HELFT. *Un mécanisme d'extraction vers C pour Why3*, in "Vingt-neuvièmes Journées Francophones des Langages Applicatifs", Banyuls-sur-mer, France, S. BOLDO, N. MAGAUD (editors), January 2018, https://hal.inria.fr/hal-01653153

[68] 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° 2, pp. 242–253, http://hal.inria.fr/inria-00533968/en/

[69] S. DE GOUW, J. ROT, F. S. DE BOER, R. BUBEL, R. HÄHNLE. *OpenJDK's Java.utils.Collection.sort() Is Broken: The Good, the Bad and the Worst Case*, D. KROENING, C. S. PĂSĂREANU (editors), Springer International Publishing, Cham, 2015, pp. 273–289

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