Activity Report 2012

Team TOCCATA

Certified Programs, Certified Tools, Certified Floating-Point Computations

RESEARCH CENTER
Saclay - Île-de-France

THEME
Programs, Verification and Proofs

Table of contents

Team TOCCATA

Keywords: Proofs Of Programs, Automated Theorem Proving, Interactive Theorem Proving, Safety, Floating-point Numbers

The Toccata team is a research team common to Inria - Saclay Île-de-France, CNRS, and Université Paris-Sud. Researchers are also members of the LRI (Laboratoire de Recherche en Informatique, UMR 8623).

Creation of the Team: September 01, 2012 .

1. Members

Research Scientists

Claude Marché [Team Leader, Senior Researcher, HdR]
Sylvie Boldo [Team Vice-Leader, Junior Researcher]
Évelyne Contejean [Junior Researcher CNRS]
Jean-Christophe Filliâtre [Junior Researcher CNRS, HdR]
Guillaume Melquiond [Junior Researcher]
Arthur Charguéraud [Junior Researcher, since Oct.]

Faculty Members

Christine Paulin-Mohring [Professor Université Paris-Sud, HdR]
Sylvain Conchon [Associate Professor Université Paris-Sud, HdR]
Andrei Paskevich [Associate Professor Université Paris-Sud]

PhD Students

François Bobot [Université Paris-Sud, Until Aug.]
Claire Dross [CIFRE Adacore]
Stefania Dumbrava [Université Paris-Sud, Since Oct.]
Paolo Herms [CEA grant until Sep., then Inria grant]
Mohamed Iguernelala [Université Paris-Sud]
Alain Mebsout [Université Paris-Sud]
Catherine Lelay [Inria Digiteo grant]
Thi-Minh-Tuyen Nguyen [Inria Digiteo grant, until June]
Asma Tafat Bouzid [Université Paris-Sud]

Post-Doctoral Fellows

Denis Cousineau [Post-doc FUI Grant Hi-Lite, until Sep.]
Cody Roux [Post-doc ANR grant, until July]

Visiting Scientist

Pierre Courtieu [Associate Professor CNAM, since Oct.]

Administrative Assistant

Régine Bricquet [TR]

Others

Levs Gondelmans [E.N.S. Paris, Master 1 intern, March to Aug.]
Nicolas Lupinski [Université Paris-Sud, Master 2 intern, March to Aug.]
Remi El Sibaie [Université Paris-Sud, Licence 3 intern, April to Jul.]

2. Overall Objectives

2.1. Introduction

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

2.1.1. State of the Art of Program Verification

The importance of software in critical systems increased a lot in the last decade. Such a software appears in various application domains like transportation (e.g. airplanes, railway), communication (e.g. smart phones) or banking. The set of available features is quickly increasing, together with the number of lines of codes 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, the main process to check safety of a software was to apply heavy test campaigns, which take a large part of the costs of software development.

This is why *static analysis* techniques were invented to complement tests. The general aim is to analyze a program code without executing it, to get as much guarantees as possible on all possible executions at once. The main classes of static analysis techniques are:

- *Abstract Interpretation*: it approximates program execution by abstracting program states into well-chosen abstraction domains. The reachable abstract states are then analyzed in order to detect possible mistakes, corresponding to abstract states that should not occur. The efficiency of this approach relies on the quality of the approximation: if it is too coarse, false positives will appear, which the user needs to analyze manually to determine if the error is real or not. A major success of this kind of approach is the verification of absence of run-time errors in the control-command software of the Airbus A380 by the tool *Astrée* [77].
- *Model-checking*: it denotes a class of approaches that got a great success in industry, e.g. the quality of device drivers of Microsoft's Windows operating system increased a lot by systematic application of such an approach [76]. A program is abstracted into a finite graph representing an approximation of its execution. Functional properties expected for the execution can be expressed using formal logic (typically temporal logic) that can be checked valid by an exploration of the graph. The major issue of model-checking is that the size of the graph can get very large. Moreover, to get less coarse approximations, one may be interested in abstracting a program into an infinite graph. In that case, extensions of model-checking are proposed: bounded model-checking, symbolic model-checking, etc. *Predicate Abstraction* is also a rather successful kind of model-checking approach because of its ability of getting iteratively refined to suppress false positives [48].
- *Deductive verification*: it differs from the other approaches in that it does not approximate program execution. It originates from the well-known *Hoare logic* approach. Programs are formally specified using expressive logic languages, and mathematical methods are applied to formally prove that a program meets its specification.

The *Toccata* project is mainly interested in exploring the deductive verification approach, although we believe that the class of approaches above are compatible. We indeed have studied some way to combine approaches in the past [10] [87][24].

2.1.2. Deductive Program Verification nowadays

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, e.g. the *Atelier B* system was used to develop part of the embedded software of the Paris metro line 14 [56] and other railroad-related systems, a formally proved C compiler was developed using the *Coq* proof assistant [92], Microsoft's hypervisor for highly secure virtualization was verified using VCC [78] and the Z3 prover

[110], 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 [89]. Another sign of recent progress is the emergence of deductive verification competitions.

In the deductive verification context, there are two main families of approaches. Methods in the first family build on top of mathematical proof assistants (e.g. Coq, Isabelle) in which both the models and the programs are encoded, and the proofs that a 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 [55], JML [67]) and automatically producing a set of mathematical formulas (the *verification conditions*) which are typically proved using automatic provers (e.g. Z3, *Alt-Ergo* [57], CVC3 [53]). 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 progresses 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. They can be applied to mainstream languages, but usually only a subset of them is supported. 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.

2.1.3. Overview of our Former Contributions

To illustrate our past contributions in the domain of deductive verification, we provide below a set of reference publications of our former scientific results, published in the past 4 years.

Concerning Automated Deduction, our references publications are: a TACAS'2011 paper [6] on an SMT decision procedure for associativity and commutativity, an IJCAR'2012 paper [20] about an original contribution to decision procedures for arithmetic, a CAV'2012 paper [24] presenting a SMT-based model checker, a FroCos'2011 paper [2] presenting an new approach for encoding polymorphic theories into monomorphic ones.

Regarding deductive program verification in general, a first reference publication is the habilitation thesis of Filliâtre [8] which provides a very good survey of our recent contributions. A shorter version appears in the STTT journal [82]. The ABZ'2012 paper [29] is a representative publication presenting an application of our *Why3* system to solving proof obligations coming from Atelier B. The VSTTE'2012 paper [27] is a reference case study publication: proof of a program solving the n-queens problem, that uses bitwise operations for efficiency.

In industrial applications, numerical calculations are very common (e.g. control software in transportation). Typically they involve floating-point numbers. Concerning the analysis of numerical programs, our representative publications are: a paper in the MCS journal in 2011 [4] presenting on various examples our approach of proving behavioral properties of numerical C programs using *Frama-C/Jessie*, a paper in the TC journal in 2010 [109] presenting the use of the Gappa solver for proving numerical algorithms, a paper in the ISSE journal in 2011 [66] together with a paper at the CPP'2011 conference [102] presenting how we can take architectures and compilers into account when dealing with floating-point programs. We also contributed to the Handbook of Floating-Point Arithmetic in 2009 [101]. 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, presented in part at ITP'2010 [61] and in another part at ICALP'2009 [3] and fully in a paper in the JAR journal [14].

Finally, about the theme of certification of analysis tools, the reference papers are: a PEPM'2010 [75] and a RTA'2011 paper [7] on certified proofs of termination and other related properties of rewriting systems, and a VSTTE'2012 paper [28] presenting a certified VC generator.

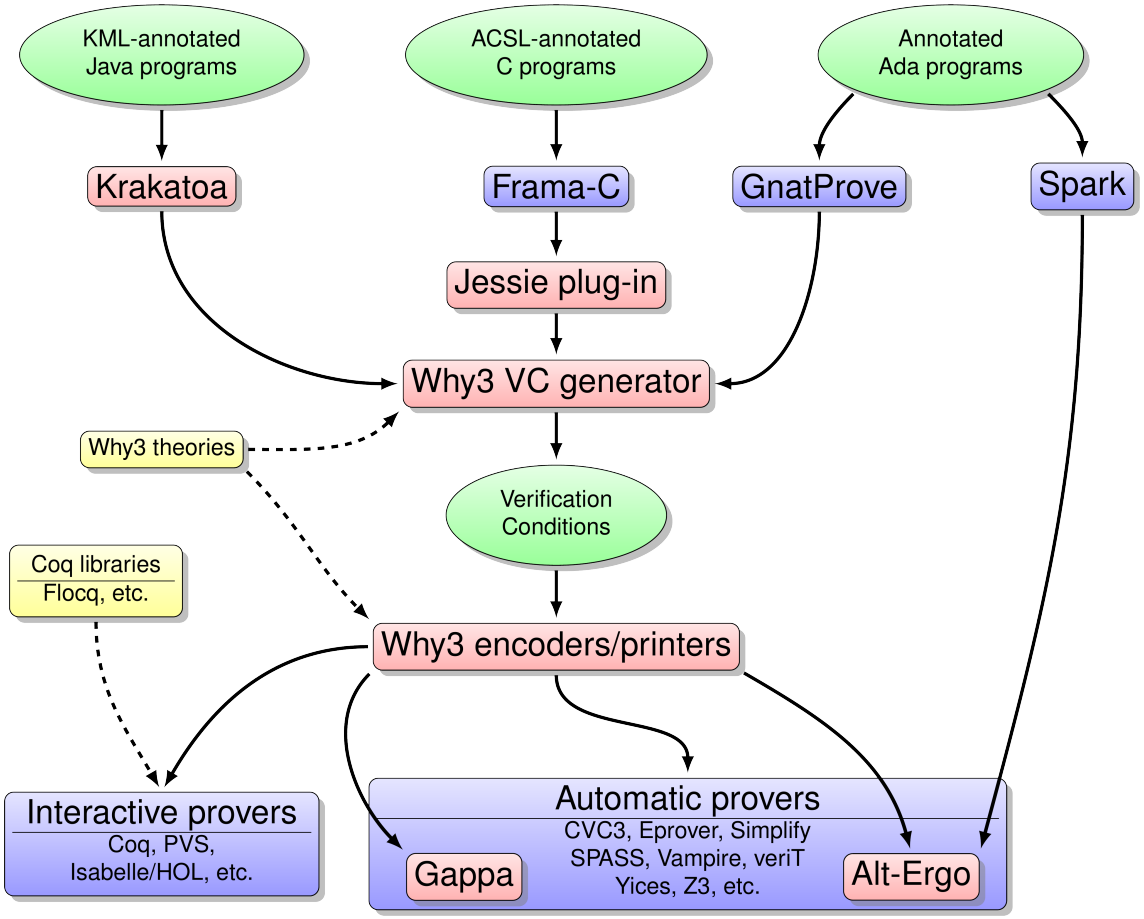

Figure 1. Interactions between our tools and with others

The diagram of Figure 1 details how our tools interact with each other and with other tools. The tools in pink boxes are designed by us, while those in blue boxes are designed by partners. The central tool is *Why3* [1][17], which includes both a Verification Condition generator and a set of encoders and printers. The VC generator reads source programs in a dedicated input language that includes both specifications and code. It produces verification conditions that are encoded and printed into various formats and syntax so that a large set of interactive or automatic provers can be used to discharge the verification conditions.

Among automated provers, our own tool *Alt-Ergo* [57] is an SMT solver that is able to deal with quantifiers, arithmetic in integers or real numbers, and a few other theories. *Gappa* [9] is a prover designed to support arithmetic on real numbers and floating-point rounding. As front-ends, our tool *Krakatoa* [94] reads annotated Java source code and produces *Why3* code. The tool *Jessie* [100][10] is a plug-in for the *Frama-C* environment (which we designed in collaboration with CEA-List); it does the same for annotated C code. The GnatProve tool is a prototype developed by Adacore company; it reads annotated Ada code and also produces *Why3* code. Spark is an ancestor of GnatProve developed by Altran-Praxis company; it has a dedicated prover but it was recently modified so as to produce verification conditions in a suitable format for *Alt-Ergo*.

Last but not least, the modeling of programs semantics and the specification of their expected behaviors is based on some libraries of mathematical theories that we develop, either in the logic language of *Why3* or in *Coq*. These are the yellow boxes of the diagram. For example, we developed in *Coq* the Flocq library for the formalization of floating-point computations [5].

2.2. Highlights of the Year

A major event in the life of our team this year is naturally its creation, as a refoundation of the former ProVal team, starting officially on September 1st, with C. Marché as a new leader. This report indeed covers all the activities of the team in 2012, including the activities of ProVal from January to August.

Another important event is the arrival of Arthur Charguéraud as a new “Chargé de Recherche”, since October.

The current section and the next one present the scientific foundations, objectives and axes of research of the new team. The theme of verification of numerical programs, that took importance in the former project, is now a major axis. We also emphasize a new axis of research concerning the certification of tools.

3. Scientific Foundations

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 originalities is our 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. This is a new goal of the team: 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 that we propose is to develop certified tools: analysis tools that are themselves formally proved correct.

As mentioned above, some of the members of the team have an internationally recognized expertise on deductive program verification involving floating-point computation [5], including both interactive proving and automated solving [9]. Indeed we noticed that the verification of numerical programs is a representative case that can benefit a lot from combining automatic and interactive theorem proving [62][4]. This is why certification of numerical programs is another axis of this proposition.

The *Toccata* project emphasizes two new axes of research: certified tools and verification of numerical programs. Additionally we want to continue the fundamental studies we conducted in the past concerning deductive program verification in general. This is why our detailed scientific programme is structured into three themes:

1. Certified Programs,
2. Certified Tools,
3. Certified Numerical Programs.

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 which is that it conforms to a rigorous process and/or a norm.

3.2. Certified Programs

This theme of research builds upon our expertise on the development of methods and tools for proving programs, from source codes annotated with specifications to proofs. In the past years, we tackled programs written in mainstream programming languages, with the system *Why3* and the front-ends *Krakatoa* for Java source code, and Frama-C/Jessie for C code. However, Java and C programming languages were designed a long time ago, and certainly not with the objective of formal verification in mind. This raises a lot of difficulties when designing specification languages on top of them, and verification condition generators to analyze them. On the other hand, we designed and/or used the *Coq* and *Why3* languages and tools for performing deductive verification, but those were not designed as programming languages that can be compiled into executable programs.

Thus, a new axis of research we propose is the design of an environment that is aimed to both programming and proving, hence that will allow to develop certified programs. To achieve this goal, there are two major axes of theoretical research that needs to be conducted, concerning on the one hand methods required to support genericity and reusability of certified components, and on the other hand the automation of the proof of the verification conditions that will be generated.

3.2.1. Genericity and Reusability of Certified Components

A central ingredient for the success of deductive approaches in program verification is the ability to reuse components that are already proved. This is the only way to scale the deductive approach up to programs of larger size. As for programming languages, a key aspect that allow reusability is *genericity*. In programming languages, genericity typically means parametricity with respect to data types, e.g. *polymorphic types* in functional languages like ML, or *generic classes* in object-oriented languages. Such genericity features are essential for the design of standard libraries of data structures such as search trees, hash tables, etc. or libraries of standard algorithms such as for searching, sorting.

In the context of deductive program verification, designing reusable libraries needs also the design of *generic specifications* which typically involve parametricity not only with respect to data types but also with respect to other program components. For example, a generic component for sorting an array needs to be parametrized by the type of data in the array but also by the comparison function that will be used. This comparison function is thus another program component that is a parameter of the sorting component. For this parametric component, one needs to specify some requirements, at the logical level (such as being a total ordering relation), but also at the program execution level (like being *side-effect free*, i.e. comparing of data should not modify the data). Typically such a specification may require *higher-order* logic.

Another central feature that is needed to design libraries of data structures is the notion of data invariants. For example, for a component providing generic search trees of reasonable efficiency, one would require the trees to remain well-balanced, over all the life time of a program.

This is why the design of reusable certified components requires advanced features, such as *higher-order specifications and programs*, *effect polymorphism* and *specification of data invariants*. Combining such features is considered as an important challenge in the current state of the art (see e.g. [90]). The well-known proposals for solving it include *Separation logic* [107], *implicit dynamic frames* [105], and *considerate reasoning* [106]. Part of our recent research activities were aimed at solving this challenge: first at the level of specifications, e.g. we proposed generic specification constructs upon Java [108] or a system of theory cloning in our system *Why3* [1]; second at the level of programs, which mainly aims at controlling side-effects to avoid unexpected breaking of data invariants, thanks to advanced type checking : approaches based on *memory regions*, *linearity* and *capability-based* type systems [69], [88], [50].

A concrete challenge that should be solved in the future is: what additional constructions should we provide in a specification language like ACSL for C, in order to support modular development of reusable software components? In particular, what would be an adequate notion of module, that would provide a good notion of abstraction, both at the level of program components and at the level of specification components.

3.2.2. Automated Deduction for Program Verification

Verifying that a program meets formal specifications typically amounts to generate *verification conditions* e.g. using a weakest precondition calculus. These verification conditions are purely logical formulas—typically in first-order logic and involving arithmetic in integers or real numbers—that should be checked to be true. This can be done using either automatic provers or interactive proof assistants. Automatic provers do not need user interaction, but may run forever or give no conclusive answer.

There are several important issues to tackle. Of course, the main general objective is to improve automation as much as possible. We want to continue our efforts around our own automatic prover *Alt-Ergo* towards more expressivity, efficiency, and usability, in the context of program verification. More expressivity means that the prover should better support the various theories that we use for modeling. Toward this direction, we aim at designing specialized proof search strategies in *Alt-Ergo*, directed by rewriting rules, in the spirit of what we did for the theory of associativity and commutativity [6].

A key challenge is also in the better handling of quantifiers. SMT solvers, including *Alt-Ergo*, deal with quantifiers with a somewhat ad-hoc mechanism of heuristic instantiation of quantified hypotheses using the so-called *triggers* that can be given by hand [42], [32]. This is completely different from resolution-based provers of the TPTP category (E-prover, Vampire, etc.) which use unification to apply quantified premises. A challenge is thus to find the best way to combine these two different approaches of quantifiers. Another challenge is to add some support for higher-order functions and predicates in this SMT context, since as said above, reusable certified components will require higher-order specifications. There are a few solutions that were proposed yet, that amount to encode higher-order goals in first-order ones [88].

Generally speaking, there are several theories, interesting for program verification, that we would like to add as built-in decision procedures in a SMT context. First, although there already exist decision procedures for variants of bit-vectors, these are not complete enough to support what is needed to reason on programs that manipulate data at the bit-level, in particular if conversions from bit-vectors to integers or floating-point numbers are involved [12]. Regarding floating-point numbers, an important challenge is to integrate in an SMT context a decision procedure like the one implemented in our tool *Gappa*.

Another goal is to improve the feedback given by automatic provers: failed proof attempts should be turned into potential counterexamples, so as to help debugging programs or specifications. A pragmatic goal would be to allow cooperation with other verification techniques. For instance, testing could be performed on unproved goals. Regarding this cooperation objective, an important goal is a deeper integration of automated procedures in interactive proofs, like it already exists in Isabelle [68]. We now have a prototype for a *Why3* tactic in *Coq* that we plan to improve.

3.2.3. An environment for both programming and proving

As said before, a new axis of research we would like to follow is to design a language and an environment for both programming and proving. We believe that this will be a fruitful approach for designing highly trustable software. This is a similar goal as projects Plaid, Trellys, ATS, or Guru, mentioned above.

The basis of this research direction is the *Why3* system, which is in fact a reimplementation from scratch of the former *Why* tool, that we started in January 2011. This new system supports our research at various levels. It is already used as an intermediate language for deductive verification.

The next step for us is to develop its use as a true programming language. Our objective is to propose a language where programs could be both executed (e.g. thanks to a compiler to, say, *OCaml*) and proved correct. The language would basically be purely applicative (i.e. without side-effects, e.g. close to ML) but incorporating specifications in its core. There are, however, some programs (e.g. some clever algorithms) where a bit of imperative programming is desirable. Thus, we want to allow some form of imperative features, but in a very controlled way: it should provide a strict form of imperative programming that is clearly more amenable to proof, in particular dealing with data invariants on complex data structures.

As already said before, reusability is a key issue. Our language should propose some form of modules with interfaces abstracting away implementation details. Our plan is to reuse the known ideas of *data refinement* [99] that was the foundation of the success of the B method. But our language will be less constrained than what is usually the case in such a context, in particular regarding the possibility of sharing data, and the constraints on composition of modules, there will be a need for advanced type systems like those based on regions and permissions.

The development of such a language will be the basis of the new theme regarding the development of certified tools, that is detailed in Section 3.3 below.

3.2.4. Extra Exploratory Axes of Research

In this theme concerning certified programs, there are a few extra exploratory topics that we plan to explore.

Concurrent Programming So far, we only investigated the verification of sequential programs. However, given the spreading of multi-core architectures nowadays, it becomes important to be able to verify concurrent programs. This is known to be a major challenge. We plan to investigate in this direction, but in a very careful way. We believe that the verification of concurrent programs should be done only under restrictive conditions on the possible interleaving of processes. In particular, the access and modification of shared data should be constrained by the programming paradigm, to allow reasonable formal specifications. In this matter, the issues are close to the ones about sharing data between components in sequential programs, and there are already some successful approaches like separation logic, dynamic frames, regions, and permissions.

Resource Analysis The deductive verification approaches are not necessarily limited to functional behavior of programs. For example, a formal termination proof typically provides a bound on the time complexity of the execution. Thus, it is potentially possible to verify resources consumption in this way, e.g. we could prove WCET (Worst Case Execution Times) of programs. Nowadays, WCET analysis is typically performed by abstract interpretation, and is applied on programs with particular shape (e.g. no unbounded iteration, no recursion). Applying deductive verification techniques in this context could allow to establish good bounds on WCET for more general cases of programs.

Other Programming Paradigms We are interested in the application of deductive methods in other cases than imperative programming à la C, Java or Ada. Indeed, in the recent years, we applied proof techniques to randomized programs [45], to cryptographic programs [49]. We plan to use proof techniques on applications related to databases. We also have plans to support low-level programs such as assembly code [81], [102] and other unstructured programming paradigm.

We are also investigating more and more applications of SMT solving, e.g. in model-checking approach (for example in Cubicle [1] [24]) or abstract interpretation techniques (new project Cafein that will start in 2013) and also for discharging proof obligations coming from other systems like Atelier B [29] (new project BWare, Section 8.2.1).

[1] http://cubicle.lri.fr/

3.3. Certified Tools

The goal of this theme is to guarantee the soundness of the tools we develop. Indeed it goes beyond that: our goal is to promote our future *Why3* environment so that *others* could develop certified tools. Tools like automated provers or program analysers are good candidate case studies because they are mainly performing symbolic computations, and as such they are usually programmed in a mostly purely functional style.

We conducted several experiments of development of certified software in the past. First, we have a strong expertise in the development of *libraries* in *Coq*: the Coccinelle library [74] formalizing term rewriting systems, the Alea library [45] for the formalization of randomized algorithms, several libraries formalizing floating-point numbers (Floats [58], Gappalib [97], and now Flocq [5] which unifies the formers). Second we recently conducted the development of a certified decision procedure [93] that corresponds to a core part of *Alt-Ergo*, and a certified verification condition generator for a language [28] similar to *Why*. On-going work aims at building, still in *Coq*, a certified VC generator for C annotated in ACSL [55], based on the operational semantics formalized in the CompCert certified compiler project [92].

To go further, we have several directions of research in mind.

3.3.1. Formalization of Binders

Using the *Why3* programming language instead of *Coq* allows more freedom. For example, it should allow one to use a bit of side-effects when the underlying algorithm justify it (e.g. hash-consing, destructive unification). On the other hand, we will lose some *Coq* features like dependent types that are usually useful when formalizing languages. Among the issues that should be studied, we believe that the question of the formalization of binders is both central and challenging (as exemplified by the POPLmark international challenge [47]).

The support of binders in *Why3* should not be built-in, but should be under the form of a reusable *Why3* library, that should already contain a lot of proved lemmas regarding substitution, alpha-equivalence and such. Of course we plan to build upon the former experiments done for the POPLmark challenge. Although, it is not clear yet that the support of binders only via a library will be satisfactory. We may consider addition of built-in constructs if this shows useful. This could be a form of (restricted) dependent types as in *Coq*, or subset types as in PVS.

3.3.2. Theory Realizations, Certification of Transformations

As an environment for both programming and proving, *Why3* should come with a standard library that includes both certified libraries of programs, but also libraries of specifications (e.g. theories of sets, maps, etc.).

The certification of those *Why3* libraries of specifications should be addressed too. *Why3* libraries for specifying models of programs are commonly expressed using first-order axiomatizations, which have the advantage of being understood by many different provers. However, such style of formalization does not offer strong guarantees of consistency. More generally, the fact that we are calling different kind of provers to discharge our verification conditions raises several challenges for certification: we typically apply various transformations to go from the *Why3* language to those of the provers, and these transformations should be certified too.

A first attempt in considering such an issue was done in [29]. It was proposed to certify the consistency of a library of specification using a so-called *realization*, which amounts to "implementing" the library in a proof assistant like *Coq*. This will be an important topic of the new ANR project BWare.

3.3.3. Certified Theorem Proving

The goal is to develop *certified* provers, in the sense that they are proved to give a correct answer. This is an important challenge since there have been a significant amount of soundness bugs discovered in the past, in many tools of this kind.

The former work on the certified core of *Alt-Ergo* [93] should be continued to support more features: more theories (full integer arithmetic, real arithmetic, arrays, etc.), quantifiers. Development of a certified prover that supports quantifiers should build upon the previous topic about binders.

In a similar way, the *Gappa* prover which is specialized to solving constraints on real numbers and floating-point numbers should be certified too. Currently, *Gappa* can be asked to produce a *Coq* proof of its given goal, so as to check *a posteriori* its soundness. Indeed, the idea of producing a trace is not contradictory with certifying the tool. For very complex decision procedures, the goal of developing a certified proof search might be too ambitious, and the production of an internal trace is a general technique that might be used as a workaround: it suffices to instrument the proof search and to develop a certified trace checker to be used by the tool before it gives an answer. We used this approach in the past for certified proofs of termination of rewriting systems [75]. This is also a technique that is used internally in CompCert for some passes of compilation [92].

3.3.4. Certified VC generation

The other kind of tools that we would like to certify are the VC generators. This will be a continuation of the on-going work on developing in *Coq* a certified VC generator for C code annotated in ACSL. We would like to develop such a generator in *Why3* instead of *Coq*. As before, this will build upon a formalization of binders.

There are various kinds of VC generators that are interesting. A generator for a simple language in the style of those of *Why3* is a first step. Other interesting cases are: a generator implementing the so-called *fast weakest preconditions* [91], and a generator for unstructured programs like assembly, that would operate on an arbitrary control-flow graph.

On a longer term, it would be interesting to be able to certify advanced verification methods like those involving refinement, alias control, regions, permissions, etc.

An interesting question is how one could certify a VC generator that involves a highly expressive logic, like higher-order logic, as it is the case of the *CFML* method [70] which allows one to use the whole *Coq* language to specify the expected behavior. One challenging aspect of such a certification is that a tool that produces *Coq* definitions, including inductive definitions and module definitions, cannot be directly proved correct in *Coq*, because inductive definitions and module definitions are not first-class objects in *Coq*. Therefore, it seems necessary to involve, in a way or another, a “deep embedding”, that is, a formalization of *Coq* in *Coq*, possibly by reusing the deep embedding developed by B. Barras [52].

3.4. Certified Numerical Programs

In recent years, we demonstrated our capability towards specifying and proving properties of floating-point programs, properties which are both complex and precise about the behavior of those programs: see the publications [65], [109], [61], [104], [64], [59], [98], [96] but also the web galleries of certified programs at our Web page [2], the Hisseo project [3], S. Boldo’s page [4], and industrial case studies in the U3CAT ANR project. The ability to express such complex properties comes from models developed in *Coq* [5]. The ability to combine proof by reasoning and proof by computation is a key aspect when dealing with floating-point programs. Such a modeling provides a safe basis when dealing with C source code [4]. However, the proofs can get difficult even on short programs, and to achieve them some automation is needed, and obtained by combining SMT solvers and *Gappa* [62], [79], [46][9]. Finally, the precision of the verification is obtained thanks to precise models of floating-point computations, taking into account the peculiarities of the architecture (e.g. x87 80-bit floating-point unit) and also the compiler optimizations [66], [102].

The directions of research concerning floating-point programs that we want to pursue are the following.

[2] http://toccata.lri.fr/gallery/index.en.html
[3] http://hisseo.saclay.inria.fr/
[4] http://www.lri.fr/~sboldo/research.html

3.4.1. Making Formal Verification of Floating-point Programs Easier

A first goal is to ease the formal verification of floating-point programs: the primary objective is still to improve the scope and efficiency of our methods, so as to ease further the verification of numerical programs. The on-going development of the Flocq library should be continued towards the formalization of bit-level manipulations and also of exceptional values (e.g. infinities). We believe that good candidates for applications of our techniques are smart algorithms to compute efficiently with floats, which operate at the bit-level. The formalization of real numbers need to be revamped too: higher-level numerical algorithms are usually built on some mathematical properties (e.g. computable approximations of ideal approximations), which then have to be proved during the formal verification of these algorithms.

Easing the verification of numerical programs also implies more automation. SMT solvers are generic provers well-suited for automatically discharging verification conditions, but they tend to be confused by floating-point arithmetic [31]. Our goal is to improve the arithmetic theories of *Alt-Ergo*, so that they support floating-point arithmetic along their other theories, if possible by reusing the heuristics developed for *Gappa*.

3.4.2. Continuous Quantities, Numerical Analysis

The goal is to handle floating-point programs that are related to continuous quantities. This includes numerical analysis programs we have already worked on [14] [61][3]. But our work is only a beginning: we were able to solve the difficulties to prove one particular scheme for one particular partial differential equation. We need to be able to easily prove this kind of programs. This requires new results that handle generic schemes and many partial differential equations. The idea is to design a toolbox to prove these programs with as much automation as possible. We wish this could be used by numerical analysts that are not or hardly familiar with formal methods, but are interested in the formal correctness of their schemes and their programs.

Another very interesting kind of programs (especially for industrial developers) are those based on *hybrid* systems, that is where both discrete and continuous quantities are involved. This is a longer term goal than four years, but we may try to go towards this direction. A first problem is to be able to specify hybrid systems: what are they exactly expected to do? Correctness usually means not going into a forbidden state but we may want additional behavioral properties. A second problem is the interface with continuous systems, such as sensors. How can we describe their behavior? Can we be sure that the formal specification fits? We may think about Ariane V where one piece of code was shamelessly reused from Ariane IV. Ensuring that such a reuse is allowed requires to correctly specify the input ranges and bandwidths of physical sensors.

Studying hybrid systems is among the goals of the new ANR project Cafein.

3.4.3. Certification of Floating-point Analyses

In coordination with our second theme, another objective is to port the kernel of *Gappa* into either *Coq* or *Why3*, and then extract a certified executable. Rather than verifying the results of the tool *a posteriori* with a proof checker, they would then be certified *a priori*. This would simplify the inner workings of *Gappa*, help to support new features (e.g. linear arithmetic, elementary functions), and make it scale better to larger formulas, since the tool would no longer need to carry certificates along its computations. Overall the tool would then be able to tackle a wider range of verification conditions.

An ultimate goal would be to develop the decision procedure for floating-point computations, for SMT context, that is mentioned in Section refsec:atp, directly as a certified program in *Coq* or *Why3*.

4. Application Domains

4.1. Application Domains

Keywords: embedded software, smartcards, avionics, telecommunication, transportation systems

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. The domains of application include

- Transportation: aeronautics, railroad, space flight, automotive
- Communications: mobile phones, smart phones, Web applications
- Financial applications, banking
- Medicine: diagnostic devices, computer-assisted surgery
- Databases with confidentiality requirements (e.g. health records, electronic voting)

Currently our industrial collaborations mainly belong the first of these domains: transportation. These include, in the context of the ANR U3CAT project (Airbus France,Toulouse; Dassault Aviation, Saint-Cloud; Sagem Défense et Sécurité):

- proof of C programs via *Frama-C*/*Jessie*/*Why* ;
- proof of floating-point programs ;
- use of the *Alt-Ergo* prover via CAVEAT tool (CEA) or *Frama-C*/WP.

In the context of the FUI project Hi-Lite, the Adacore (Paris) uses *Why3* and *Alt-Ergo* as back-endd to GnatProve, an environment for verification of Ada programs. This is applied in the domain of aerospace (Thales).

In the context of a new ANR project BWare, we investigate the use of *Why3* and *Alt-Ergo* as an alternative back-end for checking proof obligation generated by *Atelier B*, whose main applications are railroad-related software (http://www.methode-b.com/documentation_b/ClearSy-Industrial_Use_of_B.pdf, collaboration with Mitsubishi Electric R&D Centre Europe, Rennes; ClearSy, Aix-en-Provence)

Apart from the domain of transportation, the Cubicle model checker modulo theoriesbased on the *Alt-Ergo* SMT prover (collaboration with Intel Strategic Cad Labs, Hillsboro, OR, USA) can be applied to verification of concurrent programs and protocols (http://cubicle.lri.fr/).

5. Software

5.1. The CiME rewrite toolbox

Participants: Évelyne Contejean [contact], Claude Marché, Andrei Paskevich.

Keywords: Equational reasoning, Rewriting, Termination, Confluence, Completion

CiME is a rewriting toolbox. Distributed since 1996 as open source, at URL http://cime.lri.fr. Beyond a few dozens of users, CiME is used as back-end for other tools such as the TALP tool developed by Enno Ohlebusch at Bielefeld university for termination of logic programs; the MU-TERM tool (http://www.dsic.upv.es/~slucas/csr/termination/muterm/) for termination of context-sensitive rewriting; the CARIBOO tool (developed at Inria Nancy Grand-Est) for termination of rewriting under strategies; and the MTT tool (http://www.lcc.uma.es/~duran/MTT/) for termination of Maude programs. CiME2 is no longer maintained, and the currently developed version is CiME3, available at http://a3pat.ensiie.fr/pub. The main new feature of CiME3 is the production of traces for *Coq*. CiME3 is also developed by the participants of the A3PAT project at the CNAM, and is distributed under the Cecill-C license.

5.2. The Why platform

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

Keywords: Deductive verification, Java programming language, Java modeling language, Java Card, ANSI C programming language.

Criteria for Software Self-Assessment [5]: A-3, SO-4, SM-3, EM-2, SDL-5-down, OC-4.

The *Why* platform is a set of tools for deductive verification of Java and C source code. In both cases, the requirements are specified as annotations in the source, in a special style of comments. For Java (and Java Card), these specifications are given in JML and are interpreted by the *Krakatoa* tool. Analysis of C code must be done using the external *Frama-C* environment, and its Jessie plugin which is distributed in *Why*.

The platform is distributed as open source, under GPL license, at http://why.lri.fr/. The internal VC generator and the translators to external provers are no longer under active development, as superseded by the *Why3*system described below.

The *Krakatoa* and *Jessie* front-ends are still maintained, although using now by default the *Why3* VC generator. These front-ends are described in a specific web page http://krakatoa.lri.fr/. They are used for teaching (University of Evry, Ecole Polytechnique, etc.), used by several research groups in the world, e.g at Fraunhofer Institute in Berlin [86], at Universidade do Minho in Portugal [49], at Moscow State University, Russia (http://journal.ub.tu-berlin.de/eceasst/article/view/255).

5.3. The Why3 system

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

Keywords: Deductive verification

Criteria for Software Self-Assessment: A-3-up, SO-4, SM-4, EM-4, SDL-4, OC-4.

Why3 is the next generation of *Why*. *Why3* clearly separates the purely logical specification part from generation of verification conditions for programs. It features a rich library of proof task transformations that can be chained to produce a suitable input for a large set of theorem provers, including SMT solvers, TPTP provers, as well as interactive proof assistants.

It is distributed as open source, under GPL license, at http://why3.lri.fr/.

Why3 is used as back-end of our own tools *Krakatoa* and *Jessie*, but also as back-end of the GNATprove tool (Adacore company), and in a near future of the WP plugin of Frama-C. *Why3* has been used to develop and prove a significant part of the programs of our team gallery http://proval.lri.fr/gallery/index.en.html, and used for teaching (Master Parisien de Recherche en Informatique).

Why3 is used by other academic research groups, e.g. within the CertiCrypt/EasyCrypt project (http://easycrypt.gforge.inria.fr/) for certifying cryptographic programs.

5.4. The Alt-Ergo theorem prover

Participants: Sylvain Conchon [contact], Évelyne Contejean, Alain Mebsout, Mohamed Iguernelala.

Keywords: Automated theorem proving, Combination of decision procedures, Satisfiability modulo theories

Criteria for Software Self-Assessment: A-3-up, SO-4, SM-4-up, EM-4, SDL-5, OC-4.

Alt-Ergo is an automatic, little engine of proof dedicated to program verification, whose development started in 2006. It is fully integrated in the program verification tool chain developed in our team. It solves goals that are directly written in the *Why*'s annotation language; this means that *Alt-Ergo* fully supports first order polymorphic logic with quantifiers. *Alt-Ergo* also supports the standard [103] defined by the SMT-lib initiative.

It is currently used in our team to prove correctness of C and Java programs as part of the *Why* platform and the new *Why3* system. *Alt-Ergo* is also called as an external prover by the Pangolin tool developed by Y. Regis Gianas, Inria project-team Gallium http://code.google.com/p/pangolin-programming-language/. Alt-Ergo is usable as a back-end prover in the SPARK verifier for ADA programs, since Oct 2010. It is planed to be integrated in next generation of Airbus development process.

[5] self-evaluation following the guidelines (http://www.inria.fr/content/download/11783/409665/version/4/file/SoftwareCriteria-V2-CE.pdf) of the Software Working Group of Inria Evaluation Committee(http://www.inria.fr/institut/organisation/instances/commission-d-evaluation)

Alt-Ergo is distributed as open source, under the CeCILL-C license, at URL http://alt-ergo.lri.fr/.

5.5. The Cubicle model checker modulo theories

Participants: Sylvain Conchon [contact], Alain Mebsout.

Partners: A. Goel, S. Krstić (Intel Strategic Cad Labs in Hillsboro, OR, USA), F. Zaïdi (LRI, Université Paris-sud)

Keywords: Satisfiability modulo theories, model checking, array-based systems

Cubicle is an open source model checker 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.

Cubicle model-checks by a symbolic backward reachability analysis on infinite sets of states represented by specific simple formulas, called cubes. Cubicle is based on ideas introduced by MCMT (http://users.mat.unimi.it/users/ghilardi/mcmt/) from which, in addition to revealing the implementation details, it differs in a more friendly input language and a concurrent architecture. Cubicle is written in OCaml. Its SMT solver is a tightly integrated, lightweight and enhanced version of *Alt-Ergo*; and its parallel implementation relies on the Functory library.

5.6. Bibtex2html

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

Keywords: Bibliography, Bibtex format, HTML, World Wide Web.

Criteria for Software Self-Assessment: A-5, SO-3, SM-3, EM-3, SDL-5, OC-4.

Bibtex2html is a generator of HTML pages of bibliographic references. Distributed as open source since 1997, under the GPL license, at http://www.lri.fr/~filliatr/bibtex2html/. We estimate that between 10000 and 100000 web pages have been generated using Bibtex2html.

Bibtex2html is also distributed as a package in most Linux distributions. Package popularity contests show that it is among the 20% most often installed packages.

5.7. OCamlgraph

Participants: Jean-Christophe Filliâtre [contact], Sylvain Conchon.

Keywords: Graph, Library, *OCaml*.

OCamlgraph is a graph library for *OCaml*. It features many graph data structures, together with many graph algorithms. Data structures and algorithms are provided independently of each other, thanks to *OCaml* module system. OCamlgraph is distributed as open source, under the LGPL license, at http://OCamlgraph.lri.fr/. It is also distributed as a package in several Linux distributions. OCamlgraph is now widely spread among the community of *OCaml* developers.

5.8. Mlpost

Participants: Jean-Christophe Filliâtre [contact], François Bobot.

Keywords: Library, *OCaml*.

Mlpost is a tool to draw scientific figures to be integrated in LaTeX documents. Contrary to other tools such as TikZ or MetaPost, it does not introduce a new programming language; it is instead designed as a library of an existing programming language, namely *OCaml*. Yet it is based on MetaPost internally and thus provides high-quality PostScript figures and powerful features such as intersection points or clipping. Mlpost is distributed as open source, under the LGPL license, at http://mlpost.lri.fr/. Mlpost was presented at JFLA'09 [51].

5.9. Functory

Participant: Jean-Christophe Filliâtre [contact].

Keywords: Library, *OCaml*.

Functory is a distributed computing library for *OCaml*. The main features of this library include (1) a polymorphic API, (2) several implementations to adapt to different deployment scenarios such as sequential, multi-core or network, and (3) a reliable fault-tolerance mechanism. Functory was presented at JFLA 2011 [84] and at TFP 2011 [83].

5.10. The Pff library

Participant: Sylvie Boldo [contact].

Keywords: Interactive theorem proving, floating-point arithmetic. Criteria for Software Self-Assessment: A-2, SO-3, SM-3, EM-3, SDL-5, OC-4.

The Pff library for the *Coq* proof assistant is a formalization of floating-point arithmetic with high-level definitions and high-level properties [58].

It is distributed as open source, under a LGPL license, at http://lipforge.ens-lyon.fr/www/pff/, and is packaged in Debian and Ubuntu as "coq-float".

It was initiated by M. Daumas, L. Rideau and L. Théry in 2001, and then developed and maintained by S. Boldo since 2004. It is now only maintained by S. Boldo. The development has ended as this library is now subsumed by the Flocq library (see below).

5.11. The Flocq library

Participants: Sylvie Boldo [contact], Guillaume Melquiond.

Keywords: Interactive theorem proving, floating-point arithmetic.

Criteria for Software Self-Assessment: A-2, SO-3, SM-3, EM-3, SDL-4, OC-4.

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 [5]. It provides a framework for developers to formally certify numerical applications.

It is distributed as open source, under a LGPL license, at http://flocq.gforge.inria.fr/. It was first released in 2010.

5.12. The Gappa tool

Participant: Guillaume Melquiond [contact].

Keywords: Automated theorem proving, floating-point arithmetic, fixed-point arithmetic.

Criteria for Software Self-Assessment: A-3, SO-4, SM-4, EM-3, SDL-4, OC-4.

Given a logical property involving interval enclosures of mathematical expressions, Gappa tries to verify this property and generates a formal proof of its validity. This formal proof can be machine-checked by an independent tool like the *Coq* proof-checker, so as to reach a high level of confidence in the certification [79] [109].

Since these mathematical expressions can contain rounding operators in addition to usual arithmetic operators, Gappa is especially well suited to prove properties that arise when certifying a numerical application, be it floating-point or fixed-point. Gappa makes it easy to compute ranges of variables and bounds on absolute or relative roundoff errors.

Gappa is being used to certify parts of the mathematical libraries of several projects, including CRlibm, FLIP, and CGAL. It is distributed as open source, under a Cecill-B / GPL dual-license, at http://gappa.gforge.inria.fr/. Part of the work on this tool was done while in the Arénaire team (Inria Rhône-Alpes), until 2008.

5.13. The Interval package for Coq

Participant: Guillaume Melquiond [contact].

Keywords: Interactive theorem proving, interval arithmetic, floating-point arithmetic.

Criteria for Software Self-Assessment: A-3, SO-4, SM-3, EM-3, SDL-4, OC-4.

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*.

It is distributed as open source, under a LGPL license, at http://www.lri.fr/~melquion/soft/coq-interval/. Part of the work on this library was done while in the Mathematical Components team (Microsoft Research–Inria Joint Research Center).

In 2010, the Flocq library was used to straighten and fill the floating-point proofs of the Interval package.

5.14. The Alea library for randomized algorithms

Participants: Christine Paulin-Mohring [contact], Pierre Courtieu.

Keywords: Interactive theorem proving, randomized algorithms, probability

Criteria for Software Self-Assessment: A-2, SO-3, SM-2, EM-3, SDL-4, OC-4.

The ALEA library is a *Coq* development for modeling randomized functional programs as distributions using a monadic transformation. It contains an axiomatisation of the real interval $[0, 1]$ and its extension to positive real numbers. It introduces definition of distributions and general rules for approximating the probability that a program satisfies a given property.

It is distributed as open source, at http://www.lri.fr/~paulin/ALEA. It is currently used as a basis of the Certicrypt environment (MSR-Inria joint research center, Imdea Madrid, Inria Sophia-Antipolis) for formal proofs for computational cryptography [54]. It is also experimented in LABRI as a basis to study formal proofs of probabilistic distributed algorithms.

5.15. The Coccinelle library for term rewriting

Participant: Évelyne Contejean [contact].

Keywords: Interactive theorem proving, Coq, rewriting, termination certificate Coccinelle is a *Coq* library for term rewriting. Besides the usual definitions and theorems of term algebras, term rewriting and term ordering, it also models some of the algorithms implemented in the CiME toolbox, such a matching, matching modulo associativity-commutativity, computation of the one-step reducts of a term, RPO comparison between two terms, etc. The RPO algorithm can effectively be run inside *Coq*, and is used in the Color developement (http://color.inria.fr/) as well as for certifying Spike implicit induction theorems in *Coq*(Sorin Stratulat).

Coccinelle is developed by Évelyne Contejean, available at http://www.lri.fr/~contejea/Coccinelle, and is distributed under the Cecill-C license.

5.16. The Coquelicot library for real analysis

Participants: Sylvie Boldo [contact], Catherine Lelay, Guillaume Melquiond.

Keywords: Interactive theorem proving, real analysis

Criteria for Software Self-Assessment: A-2, SO-4, SM-2, EM-3, SDL-1, OC-4.

Coquelicot is a *Coq* library dedicated to real analysis: differentiation, integration, and so on. It is a conservative extension of the standard library of *Coq*, but with a strong focus on usability.

Coquelicot is available at http://coquelicot.saclay.inria.fr/.

5.17. CFML

Participant: Arthur Charguéraud [contact].

Keywords: Program verification, Interactive theorem proving, *OCaml*

Criteria for Software Self-Assessment: A-2, SO-4, SM-2, EM-3, SDL-1, OC-4. The *CFML* tool supports the verification of *OCaml* programs through interactive *Coq* proofs. The tool is made of two parts: on the one hand, a characteristic formula generator implemented as a *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.

CFML is distributed under the LGPL license, and is available at http://arthur.chargueraud.org/softs/cfml/. The tool has been initially developed by A. Charguéraud in 2010, and has been maintained and improved since by the author.

6. New Results

6.1. Proofs of (Imperative) Programs

- A. Charguéraud has extended his ICFP'11 paper [70] into a journal paper, which is currently under review. This paper describes in more details the theory of characteristic formulae and the tool *CFML*, which supports the verification of *OCaml* programs through interactive *Coq* proofs.
- J.-C. Filliâtre has verified a two lines C program (solving the N-queens puzzle) using *Why3*. This case study has been presented at VSTTE 2012 [27].
- With M .Pereira and S. Melo de Sousa (Universidade da Beira Interior, Covilhã, Portugal), J.-C. Filliâtre developed an environment for proving ARM assembly code. It uses *Why3* as an intermediate VC generator. It was presented at the Inforum conference [34] (best student paper).
- F. Bobot and J.-C. Filliâtre have presented the notion of separation predicates introduced in the PhD of F. Bobot (defended December 2011) at ICFEM 2012 [21].
- S. Conchon and A. Mesbout, in collaboration with F. Zaïdi (Fortesse team, LRI) and A. Goel and S. Krstić (Strategic Cad Labs, INTEL), have presented a tool paper about the Cubicle model checker at CAV 2012 [24]. A more detailed description of the main algorithms implemented in Cubicle will be presented during the JFLA 2013 [73].
- A significant effort was dedicated to the development of *Why3*, with 3 public releases [39], [40], [41]. Associated with this activity, we actively participe to the new trend (that emerged in 2010-2011) of construction of international program verification benchmarks and organization of program verification competitions. We participated to the joint paper that reports on the first FoVeOOS competition [23] (http://proval.lri.fr/gallery/cost11comp.en.html). J.-C. Filliâtre and A. Paskevich wrote a detailed report [33] on the 2nd competition VSTTE competition (https://sites.google.com/site/vstte2012/compet) that they organized, published in the proceedings of the COMPARE workshop. This paper describes the competition, presents the five problems that were proposed to the participants, and gives an overview of the solutions sent by the 29 teams that entered the competition.

  Our own gallery of verified programs (http://toccata.lri.fr/gallery/index.en.html) was augmented significantly, with now approximately 100 examples, classified by topics, tools, etc.

6.2. Floating-Point and Numerical Programs

- The PhD thesis of T. Nguyen was defended in June [12]. It includes an improved version of the former approach [102] that we proposed for proving floating-point programs while taking into account architecture- and compiler-dependent features, such as the use of the x87 stack in Intel micro-processors. The underlying tool analyzes the assembly code generated by the compiler. It also includes a preliminary and independent approach for proving floating-point programs involving bit-level operations.

- C. Lelay, under the supervision of S. Boldo and G. Melquiond, has worked on easing proofs of differentiability and integrability in *Coq*. The use case was the existence of a solution to the wave equation thanks to D'Alembert's formula; the goal was to automate the process as much as possible [30]. While a major improvement with respect to *Coq* standard library, this first approach was not user-friendly enough for parametric intervals. So a different approach based on the pervasive use of total functions has been experimented with [22].
- S. Boldo, F. Clément, J.-C. Filliâtre, M. Mayero, G. Melquiond and P. Weis finished the formal proof of a numerical analysis program: the second order centered finite difference scheme for the one-dimensional acoustic wave [14].
- S. Boldo has developed a formal proof of an algorithm for computing the area of a triangle, an improvement of its error bound and new investigations in case of underflow [60].
- S. Boldo, J.-H. Jourdan, X. Leroy, and G. Melquiond have extended CompCert to get the first formally verified compiler that provably preserves the semantics of floating-point programs [63].
- G. Melquiond has kept improving the floating-point and interval theories used to perform proofs by computations in *Coq* [16].

6.3. Automated Deduction

- In collaboration with Assia Mahboubi (from Typical Inria project-team), and Guillaume Melquiond, the group involved in the development of *Alt-Ergo*, implemented and proved the correctness of a novel decision procedure for quantifier-free linear integer arithmetic [20]. 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. Our algorithm is proved sound, complete, and terminating and is implemented in the *Alt-Ergo* theorem prover.
- In their LMCS journal paper [15], S. Conchon, É. Contejean and M. Iguernelala present 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. This paper extends the results presented in [72] 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.
- In [31], S. Conchon, G. Melquiond and C. Roux 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. This procedure has been implemented in *Alt-Ergo*.
- In [42], [32], C. Dross and J. Kanig from AdaCore, in collaboration with S. Conchon and A. Paskevich propose 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 our framework, and obtains in return a sound, complete and terminating solver for his theory. A prototype implementation was realized in the Alt-Ergo prover. As a case study, a feature-rich axiomatization of doubly-linked lists was proved comlpete and terminating.
- In [38], A. Paskevich in collaboration with J. Blanchette from TU München, introduced a new format in the TPTP family (http://tptp.org), called TFF1, which extends the earlier TFF0 format (many-sorted first-order logic) with rank-1 type polymorphism. The technical report presents the syntax, typing rules, and semantics, as well as a sound and complete translation from TFF1 to TFF0. The format is designed to be easy to process by existing reasoning tools that support ML-style polymorphism. It opens the door to useful middleware, such as monomorphizers and other

translation tools that encode polymorphism in FOF or TFF0. Ultimately, the hope is that TFF1 will be implemented in popular automatic theorem provers.

- A. Paskevich and J.-C. Filliâtre implemented a new *Coq* tactic that is able call an automated prover from *Coq* environment. It uses *Why3* as an intermediate tool. This new tactic brings a very significant improvement of proof automation within *Coq*. For example, the development of a certified VC generator in *Why3* made an intensive use of this tactic. The combination of automatic and interactive theorem proving was the subject of invited talks given by J.-C. Filliâtre at the workshop "Automation in Proof Assistants" [17] (satellite workshop of ETAPS 2012) and at the international workshop on Intermediate Verification Languages [18] (BOOGIE 2012, Berkeley, California, USA, July 2012).
- Together with O. Hermant (ISEP, Paris), D. Cousineau studied the cut elimination property for deduction modulo theories. They were able to show a strong relationship the syntactic cut-elimination property and the semantic construction of pre-models: they made a full semantic proof that the existence of a pre-model entails the cut elimination property for the considered theory in deduction modulo. This is published at the RTA Conference [26].
- *TLA+* is a specification language based on standard set theory and temporal logic, developed by the TLA groupe of Microsoft Research (http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html). During the first part of his post-doc, D. Cousineau finalized a work on describing how to write *TLA+* proofs and check them with *TLAPS*, the *TLA+* Proof System. It was published as a tool description at FM Conference [25].
- S. Conchon defended his *habilitation à diriger des recherches* in December 2012. The memoir [11] provides a very good and useful survey of the scientific work of the past 10 years, around the SMT solving techniques, that leaded to the tools *Alt-Ergo* and Cubicle as they are nowadays.

6.4. Certification

- P. Herms, together with C. Marché and B. Monate (CEA List), 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 [28].
- On top of the previous 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 which will be defended in January 2013.
- A. Tafat and C. Marché started experiments of development of a certified VC generator using Whyt instead of Coq. 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 nor 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 very significantly higher amount of proof automation compared to Coq [43]. This will be presented at the JFLA conference in February 2013 [95]
- The work that we started in 2011, about the use of the *Why3* environment and its back-end provers as an alternative to the built-in prover of "Atelier B", was published at the ABZ conference [29]. This work continues in the context of the new ANR project BWare.
- With J. Almeida, M. Barbosa, J. Pinto and B. Vieira (University do Minho, Braga, Portugal), J.-C. Filliâtre developed a method for certifying programs involving cryptographic methods. It uses *Why* as an intermediate language. A journal article will appear on *Science of Computer Programming* [13].
- 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. The technique for proving robustness is adapted from methods commonly used for cryptographic protocols and our work illustrates the strengths and particularities of the induced style of reasoning about probabilistic programs. This work has been presented at the conference ITP 2012 [19].

- Supervised by J. Falcou and C. Paulin during his M2 internship, N. Lupinski developed a formalisation of a skeleton language for automated generation of parallel programs. A kernel of the language has been identified, its semantics has been formalised in *Coq* where a construction is interpreted by a relation between lists of entries and lists of outputs. A transformation scheme from the skeleton language towards JOCaml programs has been proposed and proven correct with respect to the relational semantics. This work is described in [44].
- A. Charguéraud is currently working on the JsCert project (http://jscert.org), which aims at the formalization of the semantics of the JavaScript programming language (as described in *ECMAScript Language Specification, version 5.1*) and the development of a verified JavaScript interpreter. This project is joint work with Philippa Gardner, Sergio Maffeis, Gareth Smith, Daniele Filaretti and Daiva Naudziuniene from Imperial College, and Alan Schmitt and Martin Bodin from Inria Rennes - Bretagne Atlantique. As of today, the formalization already covers a substantial amount of the JavaScript language, and the verified interpreter is able to execute a number of benchmarks taken from standard JavaScript test suites.

  The formalization of the semantics of JavaScript makes use of a novel technique, called *pretty-big-step semantics*, for representing reduction rules in big-step style without suffering from a duplication of several premises accross different rules. This duplication is indeed typical in big-step semantics describing the behavior of exceptions and of divergence. The pretty-big-step semantics is described by A. Charguéraud in a paper to appear at ESOP 2013 [71].

7. Bilateral Contracts and Grants with Industry

7.1. Airbus contract

Participant: Sylvain Conchon [contact].

This 2 years support contract has started in Sep 10, between Inria and Airbus France at Toulouse. This is to support our efforts for the DO-178B qualification of Alt-Ergo.

7.2. CIFRE contract with Adacore

Participants: Claude Marché [contact], Andrei Paskevich, Claire Dross.

Jointly with the thesis of C. Dross, supervised in collaboration with the Adacore company, we established a bilateral collaboration contract, that started in January 2012 for 3 years.

The aim is to strengthen the usability of the *Alt-Ergo* theorem prover in the context of the GnatProve environment for the verification of safety-critical Ada programs [32]. A focus is made on programs involving Ada containers [80].

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. Hisseo

Participants: Sylvie Boldo [contact], Claude Marché, Guillaume Melquiond, Thi-Minh-Tuyen Nguyen.

Hisseo is a 3 and half year Digiteo project that started in September 2008 and ended in June 2012. http://hisseo.saclay.inria.fr

The Hisseo project focuses on the problems related to the treatment of floating-point computations in the compilation process, especially in the case of the compilation of critical C code [12], [46].

Partners: CEA List (Saclay), Inria Paris-Rocquencourt (Team Gallium).

8.1.2. Coquelicot

Participants: Sylvie Boldo [contact], Catherine Lelay, Guillaume Melquiond.

Coquelicot is a 3 years Digiteo project that started in September 2011. http://coquelicot.saclay.inria.fr. S. Boldo is the principal investigator of this project.

The Coquelicot project aims at creating a modern formalization of the real numbers in Coq, with a focus on practicality [30], [22]. This is sorely needed to ease the verification of numerical applications, especially those involving advanced mathematics.

Partners: LIX (Palaiseau), University Paris 13

8.1.3. Pactole

Participants: Évelyne Contejean [contact], Jean-Christophe Filliâtre.

Pactole is a 3 year Digiteo project which started in October 2009.

The Pactole project focuses on automation and formal verification for ubiquitous, large scale environments. Tasks include proof automation techniques for distributed systems, verification conditions for fault tolerant distributed systems, specification and design of fundamental services for mobile sensor networks. The principal investigator of Pactole is Xavier Urbain.

Partners: CÉDRIC (CNAM/ENSIIE), LIP6 (UPMC).

8.2. National Initiatives

8.2.1. ANR BWare

Participants: Sylvain Conchon, Évelyne Contejean, Jean-Christophe Filliâtre, Andrei Paskevich, Claude Marché.

This is a research project funded by the programme “Ingénierie Numérique & Sécurité” of the ANR. It is funded for a period of 4 years and started on September 1, 2012. http://bware.lri.fr.

It 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 guarantees of confidence. The methodology used in this project consists in 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 [29]. 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) ; Inria teams Gallium, Deducteam and Asap ; Mitsubishi Electric R&D Centre Europe, the ClearSy company that mdevelop and maintains *Atelier B* and the OCamlPro start-up.

8.2.2. ANR DECERT

Participants: Sylvain Conchon, Évelyne Contejean.

DECERT (DEduction and CERTification) is an ANR "Domaines Emergents" project. It started on January 2009 for 3 years; the coordinator is Thomas Jensen from the Lande team of IRISA/Inria Rennes.

The goal of the project DECERT is to design and implement new efficient cooperating decision procedures (in particular for fragments of arithmetics), to standardize output interfaces based on certificates proof objects and to integrate SMT provers with skeptical proof assistants and larger verification contexts such as the Rodin tool for B and the Frama-C/Jessie tool chain for verifying C programs.

The partners are: CEA List, LORIA/Inria Nancy - Grand Est, IRISA/Inria Rennes - Bretagne Atlantique, Inria Sophia Antipolis - Méditerranée, Systerel

8.2.3. ANR FOST

Participants: Sylvie Boldo [contact], Jean-Christophe Filliâtre, Guillaume Melquiond.

FOST (Formal prOofs of Scientific compuTation programs) is a 3 years ANR "Blanc" project started in January 2009 and ended in April 2012. S. Boldo is the principal investigator of this project. http://fost.saclay.inria.fr

The FOST project follows CerPAN's footprints as it aims at developing new methods to bound the global error of a numerical program. These methods will be very generic in order to prove a large range of numerical analysis programs. Moreover, FOST aims at providing reusable methods that are understandable by non-specialists of formal methods.

Partners: University Paris 13, Inria Paris - Rocquencourt (Estime).

8.2.4. ANR U3CAT

Participants: Jean-Christophe Filliâtre, Claude Marché [contact], Guillaume Melquiond, Asma Tafat, Paolo Herms.

U3CAT (Unification of Critical C Code Analysis Techniques) is a project funded by ANR within its programme "Systèmes Embarqués et Grandes Infrastructures - ARPEGE". It aims at verification techniques of C programs, and is partly a follow-up of the former CAT project. It started in January 2009 and ended in August 2012.

The main goal of the project is to integrate various analysis techniques in a single framework, and make them cooperate in a sound way. We address the following general issues:

- Verification techniques for floating-point programs;
- Specification and verification of dynamic or temporal properties;
- Combination of static analysis techniques;
- Management of verification sessions and activities;
- Certification of the tools chains for compilation and for verification.

Partners: CEA-List (Saclay, project leader), Lande team (Inria Rennes), Gallium team (Inria Rocquencourt), Dassault Aviation (Saint-Cloud), Airbus France (Toulouse), ATOS Origin (Toulouse), CNAM Cedric laboratory (Evry), CS Communication & Systems (Toulouse), Hispano-Suiza/Safran (Moissy-Cramayel).

8.2.5. ANR Verasco

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

This is a research project funded by the programme "Ingénierie Numérique & Sécurité" of the ANR. It is funded for a period of 4 years and started on January 1st, 2012. 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, it will keep 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).

8.2.6. Systematic: Hi-Lite

Participants: Claude Marché [contact], Jean-Christophe Filliâtre, Sylvain Conchon, Évelyne Contejean, Andrei Paskevich, Alain Mebsout, Mohamed Iguernelala, Denis Cousineau.

The Hi-Lite project (http://www.open-do.org/projects/hi-lite/) is a project in the SYSTEMATIC Paris Region French cluster in complex systems design and management http://www.systematic-paris-region.org.

Hi-Lite is a project aiming at popularizing formal methods for the development of high-integrity software. It targets ease of adoption through a loose integration of formal proofs with testing and static analysis, that allows combining techniques around a common expression of specifications. Its technical focus is on modularity, that allows a divide-and-conquer approach to large software systems, as well as an early adoption by all programmers in the software life cycle.

Our involvements in that project include the use of the Alt-Ergo prover as back-end to already existing tools for SPARK/ADA, and the design of a verification chain for an extended SPARK/ADA language to verification conditions, via the *Why* VC generator.

This project is funded by the french ministry of industry (FUI), the Île-de-France region and the Essonne general council for 36 months from September 2010.

8.3. European Initiatives

8.3.1. Collaborations in European Programs, except FP7

8.3.1.1. FoVeOOS

Participants: Claude Marché [contact], François Bobot, Asma Tafat.

Program: COST (European Cooperation in the field of Scientific and Technical Research, http://www.cost.esf.org/)

Project acronym: FoVeOOS (IC-0701, http://www.cost-ic0701.org/)

Project title: Formal Verification of Object-Oriented Software

Duration: May 2008 - April 2012

Coordinator: B. Beckert, University Karlsruhe, Germany

Other partners: 40 academic groups among 18 countries in Belgium, Denmark, Estonia, France, Germany, Ireland, Israel, Italy, The Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Spain, Sweden, Switzerland and United Kingdom.

Abstract: The aim of this action is to develop verification technology with the reach and power to assure dependability of object-oriented programs on industrial scale.

8.4. International Initiatives

8.4.1. Participation In International Programs

- C. Paulin is the representative of Univ. Paris-Sud for the education part of the EIT KIC ICT Labs. She contributed to the proposition of two master programs as well as the action on weaving Innovation and Entrepreneurship in Doctoral programs and the preparation of the Summer School “Imagine the future in ICT”.

8.4.2. Other International Partners

- S. Conchon has continued his collaboration with S. Krstic and A. Goel (Intel Strategic Cad Labs in Hillsboro, OR, USA) on the development of the Cubicle SMT-based model checker [24].

- J.-C. Filliâtre has collaboration with University do Minho (Braga, Portugal) on the use of *Why* as intermediate for verification of cryptographic programs [13].
- J.-C. Filliâtre has collaboration with Universidade da Beira Interior (Covilhã, Portugal) on the use of *Why* as intermediate for verification of ARM programs [34].
- Our on-going development of a verified JavaScript interpreter, described above, is an active collaboration with people from Imperial College, London, UK.

8.5. International Research Visitors

8.5.1. Visits to International Teams

- S. Conchon visited Intel Strategic Cad Labs during summer 2012.
- J.C. Filliâtre visited SRI (Menlo Park, California, USA) during summer 2012.

9. Dissemination

9.1. Scientific Animation

9.1.1. Event organization

- C. Paulin organizer of the first DigiCosme Colloquium (http://labex-digicosme.fr/Colloque2012), Sept. 12-13, École Polytechnique, France.
- C. Paulin organizer of the 4th Conference on Interactive Theorem Proving (http://itp2013.inria.fr/) that will happen in July 2013.
- C. Marché organizer of the first DigiCosme Spring School (http://labex-digicosme.fr/Spring+School+2013 whose theme is *Program Analysis and Verification* and will happen in April 2013.

9.1.2. Editorial boards

- S. Boldo is member of the editorial committee of the popular science web site *interstices*, http://interstices.info/.
- 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*.

9.1.3. Learned societies

- J.-C. Filliâtre is a member of IFIP Working Group 1.9/2.15 (Verified Software)

9.1.4. Program committees

- C. Marché, *Tool Chair* of the program committee of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013, Rome, Italy, http://www.etaps.org/2013/tacas13), part of the ETAPS joint Conference. The tool chair is responsible for the evaluation and selection of tool papers and tool demonstrations, following precise guidelines given in the call for papers. This initiative of TACAS aims at making the selection of such submissions more accurate (http://www.etaps.org/2013/tacas13/tacas13-tool-papers-menu).
- S. Conchon, member of the program committee of the 10th International SMT workshop 2012, Manchester, UK.
- C. Paulin, member of the program committee of the third and forth conferences on Interactive Theorem Proving (ITP 2012 & 2013 http://itp2013.inria.fr/).

- É. Contejean is a member of the program committees of the 23rd International Conference on "Rewriting Techniques and Applications" (RTA 2012 http://rta2012.trs.cm.is.nagoya-u.ac.jp/) and of the 24th International "Conference on Automated Deduction" (CADE-24 http://www.cade-24.info/).
- J.-C. Filliâtre is a member of the program committees of the second conference on Interactive Theorem Proving (ITP 2011), the workshop "Analyze to Compile, Compile to Analyze" (ACCA 2011), the conference Verified Software: Theories, Tools and Experiments (VSTTE 2012), the conference NASA Formal Methods (NFM 2012), and the 10th Asian Symposium on Programming Languages and Systems (APLAS 2012).

9.1.5. Invited Presentations

- S. Boldo, "Preuve de programmes d'analyse numérique", seminar of the AriC team, Lyon, January 5th.
- C. Lelay, "Improving Real Analysis in Coq: a User-Friendly Approach to Integrals and Derivatives", Coqapprox seminar (for the Tamadi ANR), Lyon, July 11th.
- C. Lelay, "Improving Real Analysis in Coq: a User-Friendly Approach to Integrals and Derivatives", LAC/LaMHA/LTP Days, Orléans, October 25th.
- G. Melquiond, "Wave Equation Numerical Resolution: a Comprehensive Mechanized Proof of a C Program", CaCoS Workshop, Grenoble, July 26th.
- J.-C. Filliâtre, "Combining Interactive and Automated Theorem Proving in Why3", Automation in Proof Assistants 2012, Tallinn, Estonia, April 2012.
- J.-C. Filliâtre, "Combining Interactive and Automated Theorem Proving using Why3 (invited tutorial)", Second International Workshop on Intermediate Verification Languages (BOOGIE 2012),Berkeley, California, USA, July 2012.

9.2. Interaction with the scientific community

9.2.1. Collective responsibilities within Inria

- S. Boldo, elected member of the Inria Evaluation Committee. She was in the committee in charge of selecting the Inria permanent researchers (CR2 and CR1).
- S. Boldo was in the committee in charge of selecting an Inria support staff (TR, *technicien de recherche*) for the Saclay finance and accounting service (SAF).
- S. Boldo, member of the CLHS, *comité local hygiène et sécurité* and member of the CLFP, *comité local de formation permanente*.
- S. Boldo, member of the committee for the monitoring of PhD students (*commission de suivi des doctorants*).
- S. Boldo, scientific head for Saclay for the MECSI group for networking about computer science popularization inside Inria.
- S. Boldo, member of the popularization committee, *comité de médiation scientifique*, of Inria.
- C. Paulin was a member of the "Commission Scientifique" (in charge of selecting PhD students, post-doc, invited researchers funded by Inria Saclay - Île-de-France) until Aug.

9.2.2. Collective responsibilities outside Inria

- A. Charguéraud is vice-president of "France-IOI", the organization in charge of the selection and the training of the French team to the International Olympiads in Informatics.
- É. Contejean and C. Marché, nominated members of the *"conseil du laboratoire"* of LRI since April 2010.

- É. Contejean, elected member of the *"Section 6 du Comité National de la Recherche Scientifique"* since September 2012.
- S. Conchon, elected member of the *"Conseil scientifique de l'Université Paris-Sud"* since June 2012.
- C. Lelay, elected member of the *"conseil du laboratoire"* of LRI since November 2011.
- C. Lelay, elected representative of the students at the Doctoral School in Computer Science at University Paris-Sud since November 2011.
- C. Marché (since April 2007) and C. Paulin (since Sep. 2010), members of the program committee of Digiteo Labs, the world-class research park in *Île-de-France* region dedicated to information and communication science and technology, http://www.digiteo.fr/.
- C. Marché and S. Boldo, members of the *"jury de l'agrégation externe de mathématiques"* as experts in computer science, since 2012.
- G. Melquiond and C. Paulin, members of the *"commission consultative de spécialistes de l'université"*, Section 27, University Paris-Sud since April 2010.
- G. Melquiond, elected officer of the IEEE-1788 standardization committee on interval arithmetic since 2008.
- G. Melquiond is an examiner for the computer science entrance exam to École Normale Supérieure since 2010.
- C. Paulin, head of the new *Digicosme* laboratory of excellence (http://labex-digicosme.fr/)
- C. Paulin, director of the Doctoral school in Computer Science at University Paris-Sud http://edips.lri.fr/, until August 2012. Now in charge of the assembly of directors of graduate schools at the Université Paris-Sud.
- C. Paulin, member of the board of the "Commission Académique Consultative" of the Initiative d'Excellence Paris-Saclay.
- C. Paulin, president of the Computer Science Department of the University Paris-Sud https://www.dep-informatique.u-psud.fr/, since February 2012.
- C. Paulin participated to the hiring committee for a senior lecturer position at in Logic of Programs at University of Göteborg in Sweden.
- C. Paulin participated to two hiring commitees in France (professor positions at Paris-Sud University and CNAM)
- J.-C. Filliâtre is *correcteur au concours d'entrée à l'École Polytechnique* (computer science examiner for the entrance exam at École Polytechnique) since 2008.
- A. Paskevich is in charge (together with C. Bastoul) of Licence professionnelle PER (L3) at IUT d'Orsay, Paris-Sud University since September 2012.

9.3. Teaching - Supervision - Juries

9.3.1. Teaching

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/: "Proofs of Programs" http://www.lri.fr/~marche/MPRI-2-36-1/ (M2), C. Marché (12h), G. Melquiond (12h), Université Paris 7, France

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/: "Automated Deduction" https://wikimpri.dptinfo.ens-cachan.fr/doku.php?id=cours:c-2-5 (M2), S. Conchon (9h), É. Contejean (10h30), Université Paris 7, France

GDR Informatique-Mathématique school for young researchers: "Arithmétique des ordinateurs et preuves formelles" [35], S. Boldo (2h), G. Melquiond (2h).

C. Paulin lecture notes on "Introduction to the *Coq* proof-assistant for practical software verification" at the LASER 2011 Summer School on Software Engineering have been published [37].

École Jeunes Chercheurs en Programmation (EJCP 2012): J.-C. Filliâtre ,"Vérification Déductive de Programmes avec Why3" (4h) [36].

Licence (DUT): "Mathématiques" (L1), "Algo Langages" (L1), C. Lelay (64h, "moniteur" position), Université Paris-Sud (IUT d'Orsay)

Licence (L2): "Mathématiques pour l'Informatique" , C. Paulin (40h), D. Cousineau (20h), Université Paris-Sud

Licence (L3): "Eléments de logique pour l'informatique", C. Paulin (50h), Université Paris-Sud, France

Licence (L3): "Programmation fonctionnelle", S. Conchon (25h), A. Tafat (50h), M. Iguernelala (50h), Université Paris-Sud, France

Licence (L3): "Projet de programmation", S. Conchon (25h), Université Paris-Sud, France

Master (M1): "Projet de compilation" A. Tafat (33h), M. Iguernelala (33h), Université Paris-Sud, France

Master (M1): "Introduction à la compilation", S. Conchon (50h), Université Paris-Sud, France

Licence: "Langages de programmation et compilation" (L3), J.-C. Filliâtre (36h), École Normale Supérieure, France

Licence (DUT): "Systèmes d'exploitation" (L1 and L2), "Réseaux" (L2), A. Paskevich (140h), Université Paris-Sud (IUT d'Orsay), France

Licence professionnelle: "Programmation concurrente" (L3), A. Paskevich (36h), Université Paris-Sud (IUT d'Orsay), France

Licence: "INF421" (L3) et "INF431" (L3), J.-C. Filliâtre (70h), École Polytechnique, France

Master (M1): "Compilation", C. Dross, Université Paris-Sud (PolyTech)

9.3.2. Supervision of internships

J.-C. Filliâtre supervised the L3 internship of Rémy El Sibaïe on "Une bibliothèque OCaml pour la combinatoire" (May–June 2012) [85].

J.-C. Filliâtre supervised the M1 internship of Levs Gondelmans on "A Simple Induction Tactic for the Why3 Platform" (April–August 2012).

C. Paulin supervised the M2 internship of Nicolas Lupinski on "formalisation and proofs of program transformations for automated parallelisation".

9.3.3. Supervision of Theses

PhD: T. Nguyen, Taking architecture and compiler into account in formal proofs of numerical programs [12], Université Paris-Sud, June 2012, S. Boldo, C. Marché

PhD in progress: M. Iguernelala, Forward and Backward Strategies in SMT solvers, since September 2009, S. Conchon, É. Contejean

PhD in progress: A. Tafat, Modular Verification of Pointer Programs, since September 2009, C. Marché

PhD in progress: P. Herms, Certification of a Tool Chain for Verification of C programs, since October 2009, C. Marché, B. Monate (CEA List). Defence scheduled on January 14th, 2013.

PhD in progress: C. Dross, Theories and Techniques for Automated Proof of programs, since January 2011, C. Marché, A. Paskevich, and industrial supervisors Y. Moy and J. Kanig from AdaCore company.

PhD in progress: A. Mebsout, SMT-based Model-Checking, since September 2011, F. Zaidi, S. Conchon

PhD in progress: C. Lelay, Real numbers for the Coq proof assistant, since October 2011, S. Boldo, G. Melquiond.

PhD in progress: S. Dumbrava, Towards data certification, since October 2012, V. Benzaken, É. Contejean

9.3.4. Juries

- S. Boldo: PhD committee of Érik Martin-Dorel "Contributions to the Formal Verification of Arithmetic Algorithms" (ENS de Lyon, LIP laboratory, September 26th, 2012)
- C. Marché: reviewer, PhD committee of Morayo Adedjouma "Requirements Engineering Process according to Automotive Standards in a Model-Driven Framework" (University Paris-Sud, LISE laboratory of CEA-List, July 12th, 2012)
- C. Marché: president of the PhD committee of Florian Noyrit "Conception Dirigée par les modèles à l'aide de langages de modélisation hétérogènes : application aux profil UML" (University Paris-Sud, LISE laboratory of CEA-List, October 25th, 2012)
- C. Marché: president of the PhD committee of Takoua Ben Rhouma Aouina "Composition des modèles de lignes de produits logiciels" (University Paris-Sud, LISE laboratory of CEA-List, November 29th, 2012)
- C. Marché: president of the PhD committee of Abderrahmane Feliachi "Semantics Based Testing for Circus" (University Paris-Sud, LRI laboratory, December 12th, 2012)
- C. Marché: reviewer, Habilitation committee of Sylvie Putot "Static Analysis of Numerical Programs and Systems" (University Paris-Sud, MeASI laboratory of CEA-List and École Polytechnique, December 13th, 2012)
- C. Paulin: reviewer, PhD committee of Thomas Braibant, "Algèbres de Kleene, Automates, Circuits, Preuve formelle, Réecriture" (Université de Grenoble, Inria Rhone-Alpes, February 17th, 2012)
- C. Paulin: reviewer, PhD committee of Sylvain Héraud, "Vérification Semi-automatique de primitives cryptographiques" (Université de Nice Sophia-Antipolis, Inria Sophia Antipolis, March 12th, 2012)
- C. Paulin: president of the PhD committee of Stéphane Glondu "Vers une certification de l'extraction de Coq", (Université Paris Diderot, Laboratory PPS, June 1st, 2012)
- C. Paulin: reviewer, Habilitation committee of David Pichardie "Toward a Verified Software Toolchain for Java" (Université de Rennes, IRISA, November 19, 2012)
- C. Paulin: president of the PhD committee of Marc Lasson "Réalisabilité et Paramétricité dans les systèmes de type purs" (Université de Lyon, LIP Laboratory, November 20, 2012)
- J.-C Filliâtre: reviewer, PhD committee of Barbara Vieira, "Formal Verification of Cryptographic Software Implementations" (Universitad do Minho, Braga, Portugal, June 25, 2012).
- J.-C Filliâtre: Habilitation committee of Etienne Lozes, "Separation Logic: Expressiveness and Copyless Message-Passing" (ENS Cachan, July 3, 2012).
- J.-C Filliâtre: reviewer, PhD committee of Cyril Cohen, "Formalized algebraic numbers: construction and first order theory" (École Polytechnique, November 20, 2012).
- J.-C Filliâtre: reviewer, Habilitation committee of Frédéric Gava "BSP, bon à toutes les sauces" (Université Paris-est, December 10, 2012).

9.4. Industrial Dissemination

- J.-C. Filliâtre and C. Marché started in 2011 a collaboration with D. Mentré at Mitsubishi Electric R&D Centre Europe (Rennes), about the use of the *Why3* environment and its back-end provers as an alternative to the built-in prover of Atelier B. This collaboration leaded first to a publication [29] and then become part of the new ANR project BWare.

- Alt-Ergo is now used in the Spark Pro toolset, developed by Altran-Praxis, for the engineering of high-assurance software. 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 (http://www.altranpraxis.com/book/, Chapter "Advanced proof tools")
- In the context of the Hi-Lite project, the Adacore company (Paris) implements a new tool GnatProve which aims at formal verification of Ada programs. They translate annotated Ada code into the *Why3* intermediate language and then use the *Why3* system to generate proof obligations and discharge them with Alt-Ergo, or other available back-end provers. GnatProve is a prototype that aims at becoming the successor of Spark (http://www.open-do.org/2012/11/30/future-version-of-spark-will-be-based-on-ada-2012/).
- S. Conchon, A. Mebsout and F. Zaidi (ForTesSe team, LRI) continued their 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).

9.5. Popularization

- Since April 2008, S. Boldo is member of the editorial committee of the popular science web site)i(: http://interstices.info/.
- S. Boldo, scientific head for Saclay for the MECSI group for networking about computer science popularization inside Inria.
- S. Boldo, member of the popularization committee, *comité de médiation scientifique*, of Inria.
- S. Boldo visited two classes for the beginning of the computer science teaching in 2nde: lycée Jean Jaurès at Chatenay-Malabry and Jacques Monod at Clamart in January 2012.
- S. Boldo is responsible for a *mission doctorale* for popularization. She is in charge of Li Gong of the LIMSI laboratory.
- S. Boldo, head for the *Fête de la science* for the LRI laboratory in 2012. The laboratory welcomed both industrials, pupils and general public: 6 different stands, 6 classes, a total of more than 250 visiting persons.
- S. Boldo and A. Charguéraud belong to the organization committee of the *Castor informatique* http://castor-informatique.fr/, an international competition to present computer science to pupils (from *6ème* to *terminale*). More than 91,000 teenagers played on the 40 proposed exercises in November 2012.
- C. Lelay and A. Tafat are involved in the *Fête de la science* 2012.

10. Bibliography

Major publications by the team in recent years

[1] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, A. PASKEVICH. *Why3: Shepherd Your Herd of Provers*, in "Boogie 2011: First International Workshop on Intermediate Verification Languages", Wrocław, Poland, August 2011, p. 53–64, http://proval.lri.fr/publications/boogie11final.pdf.

[2] F. BOBOT, A. PASKEVICH. *Expressing Polymorphic Types in a Many-Sorted Language*, in "Frontiers of Combining Systems, 8th International Symposium, Proceedings", Saarbrücken, Germany, C. TINELLI, V. SOFRONIE-STOKKERMANS (editors), Lecture Notes in Computer Science, October 2011, vol. 6989, p. 87–102, http://proval.lri.fr/publications/bobot11frocos.pdf.

[3] S. BOLDO. *Floats & Ropes: a case study for formal numerical program verification*, in "36th International Colloquium on Automata, Languages and Programming", Rhodos, Greece, Lecture Notes in Computer Science - ARCoSS, Springer, July 2009, vol. 5556, p. 91–102.

[4] S. BOLDO, C. MARCHÉ. *Formal verification of numerical programs: from C annotated programs to mechanical proofs*, in "Mathematics in Computer Science", 2011, vol. 5, p. 377–393, http://proval.lri.fr/publications/boldo11mcs.pdf.

[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, p. 243–252, http://hal.archives-ouvertes.fr/inria-00534854/.

[6] 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, p. 45-59, http://proval.lri.fr/publications/conchon11tacas.pdf.

[7] É. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS, X. URBAIN. *Automated Certified Proofs with CiME3*, in "22nd International Conference on Rewriting Techniques and Applications (RTA 11)", Novi Sad, Serbia, M. SCHMIDT-SCHAUSS (editor), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, vol. 10, p. 21–30, http://drops.dagstuhl.de/opus/volltexte/2011/3119.

[8] J.-C. FILLIÂTRE. *Deductive Program Verification*, Université Paris-Sud, December 2011, In English, http://www.lri.fr/~filliatr/hdr/memoire.pdf, Thèse d'habilitation.

[9] 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, p. 2–17.

[10] Y. MOY, C. MARCHÉ. *Modular Inference of Subprogram Contracts for Safety Checking*, in "Journal of Symbolic Computation", 2010, vol. 45, p. 1184-1211, http://hal.inria.fr/inria-00534331/en/.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] 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, http://www.lri.fr/~conchon/bib/conchon.html.

[12] T. M. T. NGUYEN. *Taking architecture and compiler into account in formal proofs of numerical programs*, Université Paris-Sud, June 2012, http://proval.lri.fr/publications/tuyennguyen12phd.pdf.

Articles in International Peer-Reviewed Journals

[13] 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.

[14] 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", 2012, p. 1–34, http://hal.inria.fr/hal-00649240/en/.

[15] 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, p. 1–29, http://www.lmcs-online.org/ojs/viewarticle.php?id=1037&layout=abstract&iid=40.

[16] G. MELQUIOND. *Floating-point arithmetic in the Coq system*, in "Information and Computation", 2012, vol. 216, p. 14–23.

Invited Conferences

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

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

International Conferences with Proceedings

[19] 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.

[20] F. BOBOT, S. CONCHON, E. 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, p. 67–81.

[21] 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://proval.lri.fr/publications/bobot12icfem.pdf.

[22] 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, p. 289–304, http://hal.inria.fr/hal-00712938.

[23] 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://proval.lri.fr/publications/bormer12foveoos.pdf.

[24] 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.

[25] D. COUSINEAU, D. DOLIGEZ, L. LAMPORT, S. MERZ, D. RICKETTS, H. VANZETTO. *TLA+ Proofs*, in "18th International Symposium on Formal Methods", D. GIANNAKOPOULOU, D. MÉRY (editors), Lecture Notes in Computer Science, Springer, 2012, vol. 7436, p. 147–154.

[26] D. COUSINEAU, O. HERMANT. *A Semantic Proof that Reducibility Candidates entail Cut Elimination*, in "23nd International Conference on Rewriting Techniques and Applications", Nagoya, Japan, A. TIWARI (editor), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, vol. 15, p. 133–148.

[27] 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, p. 83–97, http://proval.lri.fr/publications/filliatre12vstte.pdf.

[28] 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, p. 2–17, http://proval.lri.fr/publications/herms12vstte.pdf.

[29] 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, p. 238–251, http://hal.inria.fr/hal-00681781/en/.

National Conferences with Proceeding

[30] 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/.

Conferences without Proceedings

[31] 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, p. 12–21, http://smt2012.loria.fr/ .

[32] C. DROSS, S. CONCHON, J. KANIG, A. PASKEVICH. *Reasoning with Triggers*, in "SMT workshop", Manchester, UK, P. FONTAINE, A. GOEL (editors), LORIA, 2012, http://smt2012.loria.fr/ .

[33] 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://www.lri.fr/~filliatr/pub/compare2012.pdf.

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

Scientific Books (or Scientific Book chapters)

[35] S. BOLDO, G. MELQUIOND. *Arithmétique des ordinateurs et preuves formelles*, in "École des Jeunes Chercheurs en Informatique Mathématique", Rennes, France, V. BERTHÉ, C. FROUGNY, N. PORTIER, M.-F. ROY, A. SIEGEL (editors), March 2012, p. 1–30, http://hal.inria.fr/hal-00755333.

[36] J.-C. FILLIÂTRE. *Vérification déductive de programmes avec Why3*, in "Course notes EJCP 2012", June 2012, http://why3.lri.fr/ejcp-2012/ .

[37] C. PAULIN-MOHRING. *Tools for Practical Software Verification (International Summer School, LASER 2011, Revised Tutorial Lectures)*, Lecture Notes in Computer Science, Springer, 2012, vol. 7682, chap. Introduction to the Coq proof-assistant for practical software verification, http://www.lri.fr/~paulin/LASER/course-notes.pdf .

Research Reports

[38] J. C. BLANCHETTE, A. PASKEVICH. *TFF1: The TPTP typed first-order form with rank-1 polymorphism*, Tech. Univ. Munich, 2012, http://www21.in.tum.de/~blanchet/tff1spec.pdf.

[39] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, G. MELQUIOND, A. PASKEVICH. *The Why3 platform, version 0.72*, version 0.72, LRI, CNRS & Univ. Paris-Sud & Inria Saclay, May 2012, https://gforge.inria.fr/docman/view.php/2990/7919/manual-0.72.pdf .

[40] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, G. MELQUIOND, A. PASKEVICH. *The Why3 platform, version 0.73*, version 0.73, LRI, CNRS & Univ. Paris-Sud & Inria Saclay, July 2012, https://gforge.inria.fr/docman/view.php/2990/8052/manual-0.73.pdf .

[41] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, G. MELQUIOND, A. PASKEVICH. *The Why3 platform, version 0.80*, version 0.80, LRI, CNRS & Univ. Paris-Sud & Inria Saclay, October 2012, https://gforge.inria.fr/docman/view.php/2990/8186/manual-0.80.pdf .

[42] C. DROSS, S. CONCHON, J. KANIG, A. PASKEVICH. *Reasoning with Triggers*, Inria, June 2012, n^{o} RR-7986, 29, http://hal.inria.fr/hal-00703207.

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

Other Publications

[44] N. LUPINSKI, J. FALCOU, C. PAULIN-MOHRING. *Sémantiques d'un langage de squelettes*, 2012, http://www.lri.fr/~paulin/Skel .

References in notes

[45] P. AUDEBAUD, C. PAULIN-MOHRING. *Proofs of Randomized Algorithms in Coq*, in "Science of Computer Programming", 2009, vol. 74, n^{o} 8, p. 568–589, http://hal.inria.fr/inria-00431771/en/.

[46] 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, p. 127–141, http://www.lri.fr/~marche/ayad10ijcar.pdf.

[47] B. AYDEMIR, A. BOHANNON, M. FAIRBAIRN, J. FOSTER, B. PIERCE, P. SEWELL, D. VYTINIOTIS, G. WASHBURN, S. WEIRICH, S. ZDANCEWIC. *Mechanized metatheory for the masses: The POPLmark Challenge*, in "Proceedings of the Eighteenth International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2005)", Lecture Notes in Computer Science, Springer, 2005, n^{o} 3603, p. 50–65.

[48] T. BALL, R. MAJUMDAR, T. MILLSTEIN, S. K. RAJAMANI. *Automatic predicate abstraction of C programs*, in "Proceedings of the ACM SIGPLAN 2001 conference on Programming Language Design and Implementation", ACM Press, 2001, p. 203–213.

[49] M. BARBOSA, J.-C. FILLIÂTRE, J. S. PINTO, B. VIEIRA. *A Deductive Verification Platform for Cryptographic Software*, in "4th International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2010)", Pisa, Italy, Electronic Communications of the EASST, September 2010, vol. 33, http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/461.

[50] R. BARDOU. *Verification of Pointer Programs Using Regions and Permissions*, Université Paris-Sud, October 2011, http://proval.lri.fr/publications/bardou11phd.pdf.

[51] R. BARDOU, J.-C. FILLIÂTRE, J. KANIG, S. LESCUYER. *Faire bonne figure avec Mlpost*, in "Vingtièmes Journées Francophones des Langages Applicatifs", Saint-Quentin sur Isère, Inria, January 2009, http://www.lri.fr/~filliatr/ftp/publis/mlpost-fra.pdf.

[52] B. BARRAS, B. WERNER. *Coq in Coq*, 1997.

[53] 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, p. 298–302, ftp://ftp.cs.uiowa.edu/pub/tinelli/papers/BarTin-CAV-07.pdf.

[54] G. BARTHE, B. GRÉGOIRE, S. Z. BÉGUELIN. *Formal certification of code-based cryptographic proofs*, in "POPL", Savannah, GA, USA, Z. SHAO, B. C. PIERCE (editors), ACM Press, January 2009, p. 90-101.

[55] P. BAUDIN, J.-C. FILLIÂTRE, C. MARCHÉ, B. MONATE, Y. MOY, V. PREVOSTO. *ACSL: ANSI/ISO C Specification Language, version 1.4*, 2009, http://frama-c.cea.fr/acsl.html.

[56] 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, p. 369–387.

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

[58] S. BOLDO. *Preuves formelles en arithmétiques à virgule flottante*, École Normale Supérieure de Lyon, 2004, http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2004/PhD2004-05.pdf.

[59] S. BOLDO. *Kahan's algorithm for a correct discriminant computation at last formally proven*, in "IEEE Transactions on Computers", February 2009, vol. 58, n° 2, p. 220-225, http://hal.inria.fr/inria-00171497/en/.

[60] 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.

[61] 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, p. 147–162, http://hal.inria.fr/inria-00450789/.

[62] 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, p. 59–74.

[63] 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.

[64] S. BOLDO, G. MELQUIOND. *Emulation of FMA and Correctly-Rounded Sums: Proved Algorithms Using Rounding to Odd*, in "IEEE Transactions on Computers", 2008, vol. 57, n° 4, p. 462–471, http://hal.inria.fr/inria-00080427/.

[65] S. BOLDO, J.-M. MULLER. *Exact and Approximated error of the FMA*, in "IEEE Transactions on Computers", February 2011, vol. 60, n° 2, p. 157–164, http://hal.inria.fr/inria-00429617/en/.

[66] S. BOLDO, T. M. T. NGUYEN. *Proofs of numerical programs when the compiler optimizes*, in "Innovations in Systems and Software Engineering", 2011, vol. 7, p. 151–160.

[67] 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, p. 212–232, http://dx.doi.org/10.1007/s10009-004-0167-4.

[68] S. BÖHME, T. NIPKOW. *Sledgehammer: Judgement Day*, in "IJCAR", J. GIESL, R. HÄHNLE (editors), Lecture Notes in Computer Science, Springer, 2010, vol. 6173, p. 107-121.

[69] A. CHARGUÉRAUD, F. POTTIER. *Functional Translation of a Calculus of Capabilities*, in "ACM SIGPLAN International Conference on Functional Programming (ICFP)", September 2008, p. 213–224.

[70] A. CHARGUÉRAUD. *Characteristic formulae for the verification of imperative programs*, in "Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming (ICFP)", Tokyo, Japan, M. M. T. CHAKRAVARTY, Z. HU, O. DANVY (editors), ACM, September 2011, p. 418-430.

[71] A. CHARGUÉRAUD. *Pretty-Big-Step Semantics*, March 2013.

[72] 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, p. 45-59, http://proval.lri.fr/publications/conchon11tacas.pdf.

[73] S. CONCHON, A. MEBSOUT, F. ZAÏDI. *Vérification de systèmes paramétrés avec Cubicle*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013.

[74] É. CONTEJEAN. *Coccinelle, a Coq library for rewriting*, in "Types", Torino, Italy, March 2008.

[75] É. CONTEJEAN, P. COURTIEU, J. FOREST, A. PASKEVICH, O. PONS, X. URBAIN. *A3PAT, an Approach for Certified Automated Termination Proofs*, in "Partial Evaluation and Program Manipulation", Madrid, Spain, J. P. GALLAGHER, J. VOIGTLÄNDER (editors), ACM Press, January 2010, p. 63-72.

[76] B. COOK. *Static Driver Verifier*.

[77] P. COUSOT, R. COUSOT, J. FERET, L. MAUBORGNE, A. MINÉ, D. MONNIAUX, X. RIVAL. *The ASTRÉE Analyzer*, in "ESOP", Lecture Notes in Computer Science, 2005, n^{o} 3444, p. 21–30.

[78] 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, p. 429-430.

[79] M. DAUMAS, G. MELQUIOND. *Certification of bounds on expressions involving rounded operators*, in "Transactions on Mathematical Software", 2010, vol. 37, n^{o} 1, http://hal.archives-ouvertes.fr/inria-00534350/fr/.

[80] C. DROSS, J.-C. FILLIÂTRE, Y. MOY. *Correct Code Containing Containers*, in "5th International Conference on Tests and Proofs (TAP'11)", Zurich, Lecture Notes in Computer Science, Springer, June 2011, vol. 6706, p. 102–118, http://proval.lri.fr/publications/dross11tap.pdf.

[81] J.-C. FILLIÂTRE. *Formal Verification of MIX Programs*, in "Journées en l'honneur de Donald E. Knuth", Bordeaux, France, October 2007, http://www.lri.fr/~filliatr/publis/verifmix.pdf.

[82] J.-C. FILLIÂTRE. *Deductive Software Verification*, in "International Journal on Software Tools for Technology Transfer (STTT)", August 2011, vol. 13, n^{o} 5, p. 397-403, http://proval.lri.fr/publications/filliatre11sttt.pdf.

[83] J.-C. FILLIÂTRE, K. KALYANASUNDARAM. *Functory: A Distributed Computing Library for Objective Caml*, in "Trends in Functional Programming", Madrid, Spain, Lecture Notes in Computer Science, May 2011, vol. 7193, p. 65–81.

[84] J.-C. FILLIÂTRE, K. KALYANASUNDARAM. *Une bibliothèque de calcul distribué pour Objective Caml*, in "Vingt-deuxièmes Journées Francophones des Langages Applicatifs", La Bresse, France, S. CONCHON (editor), Inria, January 2011, http://www.lri.fr/~filliatr/publis/jfla-2011.pdf.

[85] J.-C. FILLIÂTRE, R. E. SIBAÏE. *Combine : une bibliothèque OCaml pour la combinatoire*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013, http://www.lri.fr/~filliatr/pub/jfla-2013.pdf.

[86] J. GERLACH, J. BURGHARDT. *An Experience Report on the Verification of Algorithms in the C++ Standard Library using Frama-C*, in "Formal Verification of Object-Oriented Software, Papers Presented at the International Conference", Paris, France, B. BECKERT, C. MARCHÉ (editors), Karlsruhe Reports in Informatics, June 2010, p. 191–204.

[87] K. KALYANASUNDARAM, C. MARCHÉ. *Automated Generation of Loop Invariants using Predicate Abstraction*, Inria, August 2011, n^{o} 7714, http://hal.inria.fr/inria-00615623/en/.

[88] J. KANIG, J.-C. FILLIÂTRE. *Who: A Verifier for Effectful Higher-order Programs*, in "ACM SIGPLAN Workshop on ML", Edinburgh, Scotland, UK, August 2009, http://www.lri.fr/~filliatr/ftp/publis/wml09.pdf.

[89] 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, p. 107–115.

[90] G. T. LEAVENS, K. R. M. LEINO, P. MÜLLER. *Specification and verification challenges for sequential object-oriented programs*, in "Formal Aspects of Computing", 2007.

[91] K. R. M. LEINO. *Efficient weakest preconditions.*, in "Information Processing Letters", 2005, vol. 93, n^{o} 6, p. 281-288.

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

[93] S. LESCUYER. *Formalisation et développement d'une tactique réflexive pour la démonstration automatique en Coq*, Université Paris-Sud, January 2011, http://proval.lri.fr/publications/lescuyer11these.pdf.

[94] C. MARCHÉ. *The Krakatoa tool for Deductive Verification of Java Programs*, January 2009, Winter School on Object-Oriented Verification, Viinistu, Estonia, http://krakatoa.lri.fr/ws/.

[95] C. MARCHÉ, A. TAFAT. *Calcul de plus faible précondition, revisité en Why3*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013.

[96] G. MELQUIOND. *De l'arithmétique d'intervalles à la certification de programmes*, École Normale Supérieure de Lyon, Lyon, France, 2006.

[97] G. MELQUIOND. *Floating-point arithmetic in the Coq system*, in "Proceedings of the 8th Conference on Real Numbers and Computers", Santiago de Compostela, Spain, 2008, p. 93–102.

[98] G. MELQUIOND, S. PION. *Formally certified floating-point filters for homogeneous geometric predicates*, in "Theoretical Informatics and Applications", 2007, vol. 41, n^{o} 1, p. 57–70.

[99] C. MORGAN. *Programming from specifications (2nd ed.)*, Prentice Hall International (UK) Ltd., 1994.

[100] Y. MOY, C. MARCHÉ. *The Jessie plugin for Deduction Verification in Frama-C — Tutorial and Reference Manual*, Inria & LRI, 2011, http://krakatoa.lri.fr.

[101] 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.

[102] 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.

[103] S. RANISE, C. TINELLI. *The Satisfiability Modulo Theories Library (SMT-LIB)*, 2006, http://smtcomp.sourceforge.net/.

[104] S. M. RUMP, P. ZIMMERMANN, S. BOLDO, G. MELQUIOND. *Computing predecessor and successor in rounding to nearest*, in "BIT", June 2009, vol. 49, n° 2, p. 419–431, http://hal.inria.fr/inria-00337537/.

[105] J. SMANS, B. JACOBS, F. PIESSENS. *Implicit Dynamic Frames: Combining Dynamic Frames and Separation Logic*, in "ECOOP 2009 — Object-Oriented Programming", S. DROSSOPOULOU (editor), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, p. 148-172.

[106] A. J. SUMMERS, S. DROSSOPOULOU. *Considerate Reasoning and the Composite Design Pattern*, in "VMCAI", G. BARTHE, M. V. HERMENEGILDO (editors), Lecture Notes in Computer Science, Springer, 2010, vol. 5944, p. 328-344.

[107] H. TUCH, G. KLEIN, M. NORRISH. *Types, Bytes, and Separation Logic*, in "Proc. 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'07)", Nice, France, M. HOFMANN, M. FELLEISEN (editors), January 2007, p. 97-108.

[108] E. TUSHKANOVA, A. GIORGETTI, C. MARCHÉ, O. KOUCHNARENKO. *Specifying Generic Java Programs: two case studies*, in "Tenth Workshop on Language Descriptions, Tools and Applications", C. BRABRAND, P.-E. MOREAU (editors), ACM Press, 2010, http://hal.inria.fr/inria-00525784/en/.

[109] 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, p. 242–253, http://hal.inria.fr/inria-00533968/en/.

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