Activity Report 2011

Project-Team PROVAL

Proofs of programs

RESEARCH CENTER
Saclay - Île-de-France

THEME
Programs, Verification and Proofs

Table of contents

Project-Team PROVAL

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

The Proval project-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).

1. Members

Research Scientists

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

Faculty Members

Christine Paulin-Mohring [Team Leader, Professor Université Paris-Sud, HdR]
Sylvain Conchon [Associate Professor Université Paris-Sud, delegation INRIA]
Andrei Paskevich [Associate Professor Université Paris-Sud]
Xavier Urbain [Associate Professor ENSIIE, Deleg. INRIA until Aug., HdR]

Technical Staff

Alain Mebsout [Junior Engineer, until August]

PhD Students

Romain Bardou [Université Paris-Sud, until Aug.]
François Bobot [Université Paris-Sud]
Claire Dross [CIFRE Adacore]
Paolo Herms [CEA grant]
Mohamed Iguernelala [Université Paris-Sud]
Alain Mebsout [Université Paris-Sud, since Sep.]
Catherine Lelay [INRIA Digiteo grant, since Oct.]
Stéphane Lescuyer [INRIA, on leave from X-Mines, until Jan.]
Thi-Minh-Tuyen Nguyen [INRIA Digiteo grant]
Asma Tafat Bouzid [Université Paris-Sud]
Wendi Urribarrí [ATER Université Paris-Sud 11 until Sep.]

Post-Doctoral Fellows

David Baelde [Post-doc CNRS]
Denis Cousineau [Post-doc FUI Grant Hi-Lite, since Sep.]
Krishnamani Kalyanasundaram [Post-doc ANR grant, until Aug.]
Evgeny Makarov [Post-doc ANR grant, May-October]
Cody Roux [Post-doc ANR grant, since May]

Visiting Scientist

Daisuke Ishii [National Institute of Informatics, Japan, since May]

Administrative Assistant

Régine Bricquet [TR]

Others

Catherine Lelay [Univ Paris 7, Master intern, May to Sep.]
Nuno Gaspar [Univ do Minho, Portugal, Master intern,until Sep.]
Shuai Yuan [Zhejiang University, China, since Oct.]

2. Overall Objectives

2.1. Introduction

Critical software applications in the domain of transportation, telecommunication or electronic transactions are put on the market within very short delays. In order to guarantee a dependable behavior, it is mandatory for a large part of the validation of the system to be done in a mechanical way.

The ProVal team addresses this question and consequently participates to the INRIA major scientific priorities: “Programming: Security and Reliability of Computing Systems”.

Our approach uses *Type Theory* as a theoretical basis, a formalism which gives a clear semantics for representing, on a computer, both computation and deduction.

Type theory is a natural formalism for the specification and proof of *higher-order functional programs*, but we also use it as the kernel for *deductive verification of imperative programs*. It serves as a support for modeling activities (e.g. pointer programs, random computations, floating-point arithmetic, semantics).

Verification conditions (VCs) generated from programs annotated with specifications can often be expressed in simple formalisms (fragments of first-order logic) and consequently be solved using *automated deduction*. Building specialized tools for solving VCs, integrating different proof technologies, in particular interactive and automated ones, are important activities in our group.

When sophisticated tools are used for analyzing safety-critical code, their reliability is an important question: in an industrial setting, there is often a certification process. This certification is based on an informal satisfaction of development rules. We believe that decision procedures, compilers or verification condition generators (VCGs) should not act as black boxes but should be themselves specified and proved, or should produce evidence of the correctness of their output. This choice is influential in the design of our tools and is also a good challenge for them.

The project develops a generic environment (*Why*) for proving programs. *Why* generates sufficient conditions for a program to meet its expected behavior, that can be solved using interactive or automatic provers. On top of this tool, we have built dedicated environments for proving C (*Frama-C/Jessie*) or Java (*Krakatoa*) programs.

With the arrival of Sylvie Boldo in 2005 and Guillaume Melquiond in 2008 as junior researchers, the team is developing a strong expertise in the area of formal verification of floating-point arithmetic.

Our research activities are detailed further, following the three themes:

- Interactive proofs of programs,
- Proof of imperative and object-oriented programs,
- Automated deduction for program proof.

Development of tools and applications is an important transversal activity for these four themes.

2.2. Highlights

A new trend emerging in 2010-2011 is the construction of international program verification benchmarks and program verification competitions. Benchmarks include the VACID0 challenges (http://vacid.codeplex.com/ [76]) and the VerifyThis collection (http://verifythis.cost-ic0701.org/). We took our part in these efforts by proposing our own gallery of verified programs (http://proval.lri.fr/gallery/index.en.html). Regarding competitions, we proposed our own solutions to the first (informal) VSTTE competition (http://proval.lri.fr/gallery/vscomp2010.en.html), we participated to the first FoVeOOS competition (Turin, Italy, Sep. 2011) and were ranked as first, ex-aequo with two other teams (http://proval.lri.fr/gallery/cost11comp.en.html) and last but not least, we indeed organized the first formal VSTTE program verification competition (November 2011, https://sites.google.com/site/vstte2012/compet).

A paper by S. Conchon, E. Contejean and M. Iguernelala [24] presenting their work on automated reasoning modulo associative-commutative theories got the best theoretical paper award of ETAPS Conferences http://www.eatcs.org/index.php/best-etaps-paper.
BEST PAPER AWARD :
[24] **Tools and Algorithms for the Construction and Analysis of Systems**. S. CONCHON, É. CONTEJEAN, M. IGUERNELALA.

3. Scientific Foundations

3.1. Interactive proofs of programs

Participants: Sylvie Boldo, Évelyne Contejean, Jean-Christophe Filliâtre, Guillaume Melquiond, Christine Paulin-Mohring.

Higher-order strongly typed programming languages such as Objective Caml help improving the quality of software development. Static typing automatically detects possible execution errors. Higher-order functions, polymorphism, modules and functors are powerful tools for the development of generic reusable libraries. Our general goal is to enrich such a software environment with a language of annotations as well as libraries for datatypes, abstract notions and associated theorems which can express logical properties of programs and ease the possibility to automatically and interactively develop proofs of correctness of the programs.

In the past, we made contributions to the *Coq* proof assistant by adding functionalities for improving the development of formally proved functional programs. A first contribution is a new method to extract OCaml modular code from *Coq* proofs (P. Letouzey PhD thesis [80], [81]). This extraction mechanism is an original feature for the *Coq* system, and has been used by several teams around the world in order to get efficient certified code [78]. Another contribution (M. Sozeau PhD thesis [91], [92]) is an extension of the *Coq* input language for building programs with strong specifications by writing only the computational part and generating separately proof obligations (which are usually solved by tactics) and also a mechanism generalizing Type Classes à la Haskell which gives overloading in programs and proofs and facilitates the development of generic tactics..

We are using the capability of the *Coq* system to model both computation and deduction in order to explore different classes of applications. These examples involve the development of large reusable *Coq* libraries and suggest domain-specific specification and proof strategies.

3.1.1. Randomized algorithms

C. Paulin in collaboration with Ph. Audebaud from ENS Lyon, proposed a method for modeling probabilistic programs in *Coq* [49]. The method is based on a monadic interpretation of probabilistic programs as probability measures. A large *Coq* library has been developed and made publicly available (see also Section 5.12). D. Baelde has been using this library to formally prove the security of Watermarking algorithms (see also section 6.1).

3.1.2. Floating-point programs

Many industrial programs (weather forecasts, plane trajectories, simulations...) use floating-point computations, typically double precision floating-point numbers [93]. Even if each computation is as good as it can be (except for elementary functions like sine, or exponential), the final result may be very wrong with no warnings, or the program will produce unexpected behaviors (like division by zero). This is the reason why guarantees should be provided to the user. We mean to guarantee for example that, for all or part of the possible inputs, the result obtained is correct (or near enough) and that no exceptional behavior will occur [55].

A high level of guarantee is obtained by formal proofs in *Coq*. We maintain and develop large *Coq* libraries for floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties. It provides a framework for developers to formally certify numerical applications. A new such library is described in Section 5.9.

3.1.3. *Certification of tools*

Certifying the result of tools for analysing programs is a good challenge in the domain of proofs of higher-order functional programs. We obtained several results concerning formal proofs in *Coq* corresponding to automated deduction. These results are described in Section 3.3.

A PhD thesis started in Sep. 2009 has for main objective the development of a certified version the Frama-C/Jessie/Why verification chain.

3.2. Proof of Imperative and Object-Oriented programs

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Claude Marché, Tuyen Nguyen, Andrei Paskevich, Christine Paulin-Mohring, Asma Tafat, Wendi Urribarrí.

A foundation step of the project is the PhD thesis of Jean-Christophe Filliâtre [6] that proposes to establish soundness of a program with imperative features (assignments, while loops, but also exceptions and exception handlers) by means of a translation into an equivalent purely functional program with logical annotations. Such an annotated functional program is very well-suited to be expressed in *Coq*'s type theory, hence this approach allowed for the first time to prove imperative programs with *Coq* [68].

Following this thesis, a new tool called *Why* was developed. It takes as input an imperative program and a specification that this program is expected to fulfil. It produces on one hand a set of *verification conditions* (VCs): logical formulas which have to be proved in the *Coq* system ; and on the other hand a *Coq*-term which contains a functional translation of the imperative program and a proof of correctness of this program based on the VCs. It was early remarked that this tool was independent of *Coq*, because the VCs can be validated in other interactive tools or with automatic provers. This multi-prover architecture is a powerful feature of *Why*: it spreads this technology well beyond the *Coq* community.

3.2.1. *The Why platform*

Since 2002, we tackle programs written in mainstream programming languages. We first considered Java source code annotated with JML (Java Modeling Language). This method was implemented in a new tool called *Krakatoa* [10]. The approach is based on a translation from annotated Java programs into the specific language of *Why*, we then can reuse *Why*'s VCG mechanism and choose between different provers for establishing these VCs. From 2003, we followed the same approach for programs written in ANSI C [7].

The combination of the *Why* VC generator and the front-ends dealing with C or Java form a tool box for program verification, called the *Why* platform. Its overall architecture is shown on Figure 1. Nowadays, the front-end for C is in fact integrated in the Frama-C environment for static analysis of C programs (http://www.frama-c.cea.fr/), which was developed by the CEA-List in collaboration with us. Frama-C has an open architecture, structured as plugins around a shared kernel, and deductive verification of C code can be done using Why via the Jessie plugin. The annotation language for C source is also designed in collaboration with CEA, and called ACSL [54].

The central issue for the design of our platform is the modeling of memory heap for Java and C programs, handling possible aliasing (two different pointer or object expressions representing the same memory location): the *Why* VC generator does not handle aliasing by itself, indeed it does not support any form of complex data structures like objects, structures, pointers. On the other hand, it supports declaration of a kind of algebraic specifications: abstract data types specified by first-order functions, predicates and axioms. As a consequence, there is a general approach for using *Why* as a target language for *programming the semantics* of higher-level programming languages [85]. The *Krakatoa* and the Jessie memory models are inspired by the 'component-as-array' representation due to Bornat, following an old idea from Burstall, and commonly used to verify pointers programs [58]. Each field declaration f in a Java class or a C structure introduces a *Why* variable M_f in the model, which is a map (or an array) indexed by addresses. We extended this idea to handle Java arrays and JML annotations [10] and pointer arithmetic in C [7].

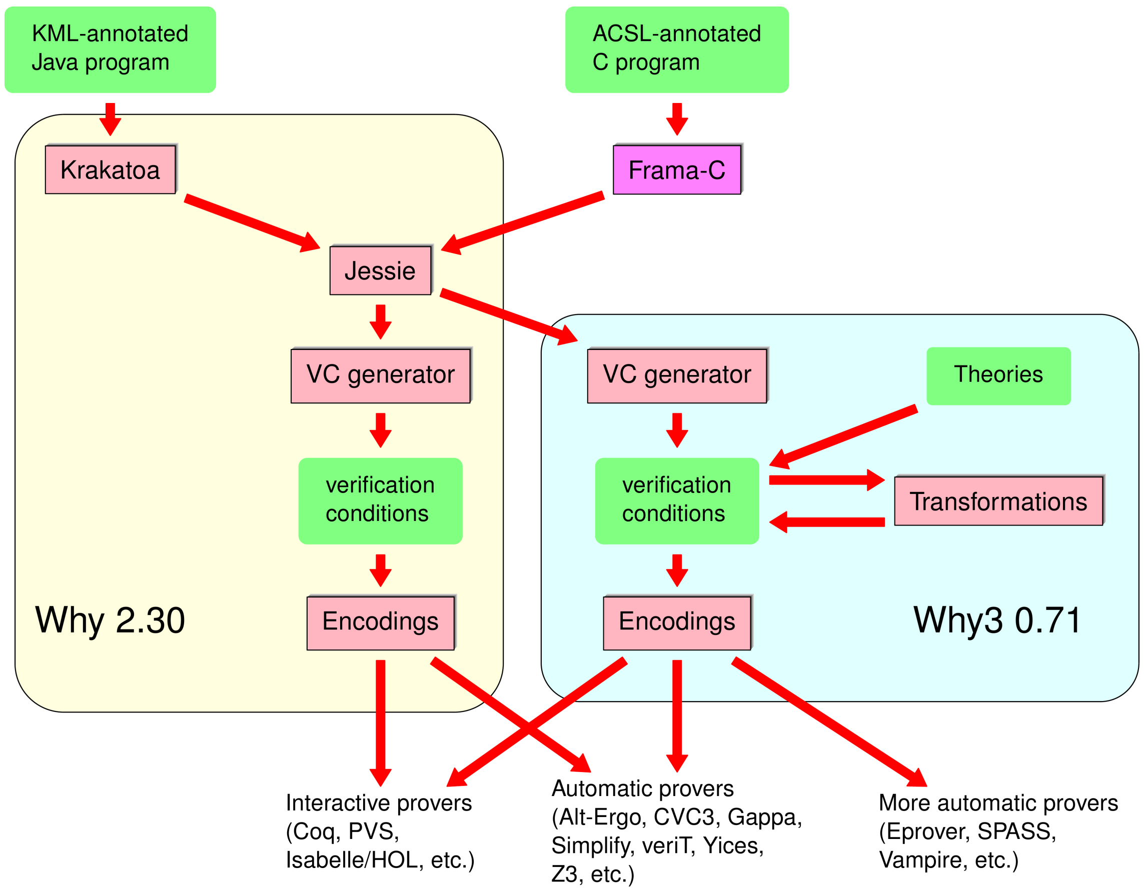
KML-annotated Java program
ACSL-annotated C program
Krakatoa
Frama-C
Jessie
VC generator
VC generator
Theories
verification conditions
verification conditions
Transformations
Why 2.30
Encodings
Encodings
Why3 0.71
Interactive provers (Coq, PVS, Isabelle/HOL, etc.)
Automatic provers (Alt-Ergo, CVC3, Gappa, Simplify, veriT, Yices, Z3, etc.)
More automatic provers (Eprover, SPASS, Vampire, etc.)

Figure 1. Architecture of certification chains: Frama-C, Why, Why3 and back-end provers

An important difficulty with programs handling pointers is to specify side-effects of a function or a method. The annotation languages offer the *assigns* clauses in specifications in order to delimitate the part of memory which is modified by a function or a method. We proposed an original modeling for such clauses [82] [7].

This kind of memory model does not scale up well for large programs. We designed an improved modeling of memory heap incorporating ideas from static analysis of memory separation, and from Reynolds' separation logic. Experiments on a C code proposed by Dassault Aviation were successful [74], [73].

The use of *Why* as intermediate language opens interesting new approaches for reasoning on programs. We studied the specification of global properties, by reuse of the validation term of *Why* in order to define a model of each function, and then express and prove properties of functions composition. Such an approach was investigated by J. Andronick in the framework of proofs of security properties on smart cards [47], [46]. We also proposed a way to handle the Java Card transaction mechanism (a specificity of Java Card memory with both persistent and volatile parts), by indeed generating a *Why* model *on-the-fly* for each Java Card applet [83], thanks again to the flexibility of the approach using *Why* as an intermediate language.

3.2.2. *Applications and case studies*

The techniques we are developing can be naturally applied in domains which require to develop critical software for which there is a high need of certification.

The *Krakatoa* tool was successfully used for the formal verification of a commercial smart card applet [75] proposed by Gemalto. This case study have been conducted in collaboration with LOOP and Jive groups. Banking applications are concerned with security problems that can be the confidentiality and protection of data, authentication, etc. The translation of such specifications into assertions in the source code of the program is an essential problem. We have been working on a Java Card applet for an electronic purse Demoney [59] developed by the company Trusted Logic for experimental purpose. Other Java Card case studies have been conducted in collaboration with Gemalto by J. Andronick and N. Rousset, in particular on global properties and Java Card transactions [47], [83].

To illustrate the effectiveness of the approach on C programs, T. Hubert and C. Marché performed a full verification of a C implementation of the Schorr-Waite algorithm [8], using *Coq* for the proofs. This is an allocation-free graph-marking algorithm used in garbage collectors, which is considered as a benchmark for verification tools. Other industrial case studies have been investigated by T. Hubert (with Dassault Aviation) [73] and by Y. Moy (with France Telecom) [88], [87].

Since the beginning of 2011, we propose on the web a *Gallery of Verified programs* (http://proval.lri.fr/gallery/index.en.html) which provides a large set of examples of programs that we proved correct, using various techniques. The gallery can be browsed using different criteria: by topics, by reference to benchmarks, by tools.

3.3. Automated deduction

Participants: Sylvain Conchon, Évelyne Contejean, Claire Dross, Mohamed Iguernelala, Stéphane Lescuyer, Claude Marché, Alain Mebsout, Andrei Paskevich, Xavier Urbain.

Our group has a long tradition of research on automated reasoning, in particular on equational logic, rewriting, and constraint solving. The main topics that have been under study in recent years are termination proofs techniques, the issue of combination of decision procedures, and generation of proof traces. Our theoretical results are mainly materialized inside our two automated provers CiME and *Alt-Ergo*.

3.3.1. *Termination*

On the termination topic, we have studied new techniques which can be automated. A fundamental result of ours is a criterion for checking termination *modularly* and *incrementally* [94], and further generalizations [84]. These criteria and methods have been implemented into the CiME2 rewrite toolbox [63]. Around 2002, several projects of development of termination tools arose in the world. We believe we have been pioneer in this growth, and indeed we organized in 2004 the first competition of such tools.

A direction of research on termination techniques was also to apply our new approaches (for rewriting) to other computing formalisms, first to Prolog programs [89] and then to membership equational programs [67], a paradigm used in the *Maude* system [45].

3.3.2. Decision Procedures

3.3.2.1. Combination

Our research related to combination of decision procedures was initiated by a result [70] obtained in collaboration with Shankar's group at SRI-international who develops the PVS environment, showing how decision procedures for disjoint theories can be combined as soon as each of them provides a so-called "canonizer" and a "solver". Existing combination methods in the literature are generally not very well understood, and S. Conchon had a major contribution, in collaboration with Sava Krstić from OGI School of Science and Engineering (Oregon Health and Science University, USA), which is a uniform description of combination of decision procedures, by means of a system of inference rules, clearly distinguished from their strategy of application, allowing much clearer proofs of soundness and completeness [9], [60].

3.3.2.2. Polymorphic Logics

In the specific domain of program verification, the goals to be proved are given as formulae in a polymorphic multi-sorted first-order logic. Some of the sorts, such as integers and arrays, are built-in as they come from the usual data-types of programming languages. Polymorphism is used as a convenience for defining the memory models of C and Java programs and is handled at the level of the *Why* tool.

In order to be able to use all the available automated theorem provers (Simplify, SMT provers), including those which handle only untyped formulae (Simplify), one has to provide a way to get rid of polymorphism.

S. Conchon and É. Contejean have proposed an encoding of polymorphic multi-sorted logic (PSL) into unsorted logic based on term transformation, rather than addition of sort predicates which was used till then. This approach was extended further by S. Lescuyer [79], J.-F. Couchot [65], N. Stouls [64].

3.3.2.3. The Alt-Ergo theorem prover

It would be more convenient to deal with polymorphism directly in the theorem prover. There was no such prover available at the beginning of 2006, that is why S. Conchon and É. Contejean decided to develop a new tool called *Alt-Ergo* which is dedicated to the resolution of polymorphic and multi-sorted proof obligations and takes as input the *Why* syntax. In 2011, *Alt-Ergo* is still the only existing prover dealing with parametric polymorphism.

Alt-Ergo is based on CC(X) [3], a generic congruence closure algorithm developed in the team, for deciding ground formulas in the combination of the theory of equality with uninterpreted symbols and an arbitrary built-in solvable theory X. Currently, CC(X) can be instantiated by the empty equational theory, by the linear arithmetics and the theory of constructors.

Alt-Ergo contains also a Fourier-Motzkin decision procedure for linear arithmetics inequalities, a home-made SAT-solver and an instantiation mechanism.

The architecture of *Alt-Ergo* is modular: each part is described by a small set of inference rules and is implemented as an OCaml functor. Moreover, the code is short ($\sim$ 10000 lines).

3.3.3. Automated proofs and certificates

A common issue to both termination techniques and decision procedures is that automatic provers use complex algorithms for checking validity of formula or termination of a computation, but when they answer that the problem is solved, they do not give any more useful information. It is highly desirable that they give a *proof trace*, that is some kind of certificate that could be double-checked by a third party, such as an interactive proof assistant like *Coq*. Indeed *Coq* is based on a relatively small and stable kernel, so that when it checks that a proof is valid, it can be trusted. Morevoer, a subpart of Coq has been proven correct in Coq [52].

3.3.3.1. Coccinelle and CiME's traces

In addition to efficient termination techniques, CiME implements in particular a semi-decision procedure for the equality modulo a set of axioms, based on ordered completion. In 2005, the former human readable proof traces have been replaced by *Coq* certificates, based on reified proof objects for a FOL logic modelled inside *Coq* [61].

É. Contejean, A. Paskevich, X. Urbain and the Cédric participants of the A3PAT project, Pierre Courtieu, Olivier Pons (CNAM), and Julien Forest, (ENSIIE) develop the new version of the CiME tool, CiME 3, associated with a *Coq* library called Coccinelle developed by É. Contejean. A trace generator outputs a trace for *Coq* in the unified framework provided by the Coccinelle library [62][4]. Coccinelle contains the corresponding modelling of terms algebras and rewriting statements, and also some generic theorems which are needed for establishing a rewriting property from a trace. For example, in order to produce a certificate of termination for a rewriting system, one may provide as a trace an ordering that contains the rewrite system, but it is also needed to have a proof that this ordering is well-founded. Such a proof (for RPO for instance) is part of Coccinelle as a generic property. Coccinelle also contains as generic theorems some powerful criteria of termination: dependency pairs [48], the main modularity theorem for termination presented in the thesis of Urbain [94] as well as innermost termination, dependency pairs for it and its equivalence with standard termination in some specific cases [72].

The main improvement over the previous approach [61] is that the *Coq* development is parameterized with respect to the equality predicate (instead of using the *Coq* native equality). This allows to deal uniformly with equality modulo a set of axioms, with termination of a set of rewrite rules, and with rewriting modulo a set of equations, such as associativity-commutativity.

Certifying termination proofs gained interest in the term rewriting community. Groups are either developing their own certifier, or producing traces for other's, thanks to a shared XML format. Since 2007, the termination competition has a category for certified termination proofs.

Further note that our efforts are not limited to termination proofs, and to date CiME 3 is the only tool able to prove and to certify *confluence* of term rewriting systems [25].

3.3.3.2. The ergo tactics

In his thesis [14], S. Lescuyer proposed new automation capabilities for the Coq proof assistant. He obtains this mechanization via an integration into Coq of decision procedures for propositional logic, equality reasoning and linear arithmetic which make up the core of the Alt-Ergo SMT solver. This integration is achieved through the reflection technique, which consists in implementing and formally proving these algorithms in Coq in order to execute them directly in the proof assistant. Because the algorithms formalized in Coq are exactly those in use in Alt-Ergo's kernel, this work significantly increases our trust in the solver. In particular, it embeds an original algorithm for combining equality modulo theory reasoning, called CC(X) and inspired by the Shostak combination algorithm, and whose justification is quite complex.

The Coq implementation of S. Lescuyer is available in the form of tactics which allow one to automatically solve formulae combining propositional logic, equality and arithmetic. In order to make these tactics as efficient as may be, he has taken special care with performance in his implementation, in particular through the use of classical efficient data structures, which we provide as a separate library.

4. Application Domains

4.1. Panorama

Many systems in telecommunication, banking or transportation involve sophisticated software for controlling critical operations. One major problem is to get a high-level of confidence in the algorithms or protocols that have been developed inside the companies or by partners.

Many smartcards in mobile phones are based on a (small) Java virtual machine. The card is supposed to execute applets that are loaded dynamically. The operating system itself is written in C, it implements security functions in order to preserve the integrity of data on the card or to offer authentication mechanisms. Applets are developed in Java, compiled, and then the byte-code is loaded and executed on the card. Applets or the operating systems are relatively small programs but they need to behave correctly and to be certified by an independent entity.

If the user expresses the expected behavior of the program as a formal specification, it is possible for a tool to check whether the program actually behaves according to the requirements.

Avionics or more generally transportation systems are another area were there are critical algorithms involved, for instance in Air Traffic control. We have collaborations in this domain with Dassault-Aviation and National Institute of Aerospace (NIA, Hampton, USA). Since 2011, we started a new collaboration with Mitsubishi Electric R&D Centre Europe (Rennes), on the construction of certified software for railroad transportation. We also recently started a collaboration with Adacore for a new environment for proving Ada source code, which has applications in transportation systems including aerospace.

5. Software

5.1. The CiME rewrite toolbox

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

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], Romain Bardou, François Bobot, Jean-Christophe Filliâtre, Guillaume Melquiond, Andrei Paskevich.

Criteria for Software Self-Assessment [1]: 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 *Why*3 system described below.

The *Krakatoa* and *Jessie* front-ends are still maintained, although using now by default the *Why*3 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 [71], and at Universidade do Minho in Portugal [50].

[1] 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)

5.3. The Why3 system

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

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

*Why*3 is the next generation of *Why*. *Why*3 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/.

*Why*3 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. *Why*3 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 will be used soon for teaching (Master Parisien de Recherche en Informatique).

5.4. The Alt-Ergo theorem prover

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

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 [90] 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 *Why*3 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.

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

5.5. Bibtex2html

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

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.

5.6. OCamlgraph

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

OCamlgraph is a graph library for Objective Caml. 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.

5.7. Mlpost

Participants: Jean-Christophe Filliâtre [contact], Stéphane Lescuyer, Romain Bardou, François Bobot.

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 Objective Caml. 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].

5.8. Functory

Participants: Jean-Christophe Filliâtre [contact], Kalyan Krishnamani.

Functory is a distributed computing library for Objective Caml. 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 [31] and at TFP 2011 [27].

5.9. The Flocq library

Participants: Sylvie Boldo [contact], Guillaume Melquiond.

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 [23]. 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.

5.10. The Gappa tool

Participant: Guillaume Melquiond [contact].

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 [66] [19].

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.

5.11. The Interval package for Coq

Participant: Guillaume Melquiond [contact].

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.

5.12. The Alea library for randomized algorithms

Participants: Christine Paulin-Mohring [contact], David Baelde.

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 [53]. It is also experimented in LABRI as a basis to study formal proofs of probabilistic distributed algorithms.

5.13. The Coccinelle library for term rewriting

Participant: Évelyne Contejean [contact].

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.

6. New Results

6.1. Models of Programming

- 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 [37]. This is a first step towards a certified VC generator for C programs annotated with ACSL [54].
- Until now, we only considered the proof of sequential programs. However the rely/guarantee approach give a natural way to extend deductive verification to concurrent programs. During his internship supervised by C. Paulin, N. Gaspar explored this idea. He formalised in Coq the axiomatic semantics of a simple concurrent language using the rely-guarranty approach [43] and proposed a translation of this language into *Why* programs in order to automatically generate the proof obligations.
- W. Urribarrí, together with C. Paulin, proposed an extension of the *Why* language with modules and functors together with a refinement calculus in order to organise large developments in a structured and abstract way. She built a prototype implementation of this calculus.

- D. Baelde, in cooperation with P. Courtieu (CNAM), D. Gross-Amblard (U. Bourgogne and Rennes), C. Paulin and X. Urbain proposed a formal proof of security for two watermarking algorithms. The proof uses a reduction of an arbitrary attack unmarking the data to the discovery of a secret key. It has been fully formalized in Coq using the ALEA library. This work has been presented at the conference TYPES 2011 and a paper is in preparation.
- Generating multimedia streams, such as in a netradio, is a task which is complex and difficult to adapt to every users' needs. D. Baelde, in cooperation with R. Beauxis (Tulane University, LA, USA) and S. Mimram (CEA List) introduce a novel approach, based on a dedicated high-level functional programming language, called Liquidsoap, for generating, manipulating and broadcasting multimedia streams [20]. Unlike traditional approaches, which are based on configuration files or static graphical interfaces, it also allows the user to build complex and highly customized systems. This language is based on a model for streams and contains operators and constructions, which make it adapted to the generation of streams. The interpreter of the language also ensures many properties concerning the good execution of the stream generation.

6.2. Proofs of Imperative Programs

- The Why3 reimplementation of the *Why* platform, started in 2010, was publicly released in 2011 [35]. The main developers are A. Paskevich, J.-C. Filliâtre, F. Bobot, and C. Marché. The language of *Why*, both programming and annotation parts, was significantly extended: algebraic types, records, pattern matching, recursive logical definitions are now supported. These logical declarations are structured in modules (a.k.a. theories). The module language comes with an original mechanism for reusing theories in specialized contexts using partial instantiations. These new features have been presented at the first international workshop on intermediate verification languages (BOOGIE 2011) [21] and will be presented at VSTTE 2012 [69].
- A. Tafat and C. Marché used the Why3 system to perform a complete proof of the "Binary Heaps" challenge [41] from the VACID-0 international collection [76]. Solving this challenge is a case study for a general approach of abstract interfaces for C programs, currently under development by A. Tafat, based on initial ideas described together with S. Boulmé which were published in a 2011 in a special issue [29].
- In 2011, we set up a web gallery to publicly expose the programs that we specified and proved. This is the so-called ProVal collection of verified programs, and available at URL http://proval.lri.fr/gallery/index.en.html.
- K. Krishnamani and C. Marché proposed a technique for automatically generating loop invariants in C programs. It is based on the well-known predicate abstraction approach, which is adapted to take into account pre-existing specifications, and to proceed modularly, that is each function of a program is processed independently, with its own sets of predicates. The approach is also extended in order to generate universally quantified invariants [38]. The prototype is available as a Frama-C plugin at URL http://proval.lri.fr/agen.
- C. Dross, together with Y. Moy (Adacore) and J.-C. Filliâtre, addressed the problem of verifying programs involving *containers*. Containers such as lists, vectors, sets or maps are an attractive alternative to ad-hoc data structures based on pointers. C. Dross gave a definition of containers whose aim is to facilitate their use in certified software, using modern proof technology and novel specification languages. Correct usage of containers and user-provided correctness properties can be checked either by execution during testing or by formal proof with an automatic prover. It relies on a formal semantics for containers and an axiomatization of this semantics targeted at automatic provers. C. Dross proved in Coq that the formal semantics is consistent and that the axiomatization thereof is correct. This work was presented at TAP 2011 [26].
- Proving that pointer programs preserve data invariants, in a modular way, is known to be a challenge. R. Bardou and C. Marché designed a new approach based on advanced static typing systems (based

on memory regions and permissions) to control memory updates and ownership of data [11]. A prototype implementation is built, called Capucine, available for download at http://romain.bardou.fr/capucine. To demonstrate the ability of this approach, the challenge "sparse arrays" of the VACID-0 benchmarks [76] has been certified [30].

- Separation logic has shown to be an elegant way to deal with programs which use data-structures with pointers. However it requires a specific logical language, provers, and specific reasoning techniques. In his PhD. F. Bobot introduced a technique to express ideas from separation logic in the traditional framework of deductive verification [12]. He proposed to derive "separation predicates" from user-supplied inductive definitions. These predicates come with suitable axiomatization, including frame rules, expressed in usual first-order logic. This translation takes special care to ensure the best use of automated theorem provers.

6.3. Automated Deduction

- In his thesis [14], S. Lescuyer formalized and designed purely reflexive tactics for automated deduction in Coq.
- É. Contejean, together with Pierre Courtieu, Julien Forest, Olivier Pons and Xavier Urbain (Cedric Laboratory, CNAM & ENSIIE) presented the last version of the rewriting toolkit CiME3 at RTA 2011 [25]. Amongst other original features, this version enjoys two kinds of engines: to handle and discover proofs of various properties of rewriting systems, and to generate Coq scripts from proof traces given in certification problem format in order to certify them with a skeptical proof assistant like Coq. Thus, these features open the way for using CiME3 to add automation to proofs of termination or confluence in a formal development in the Coq proof assistant.
- In their TACAS paper [24], 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.
- F. Bobot and A. Paskevich studied translation from a first-order logic with polymorphic types à la ML (which is the base logic of the *Why* platform and the *Alt-Ergo* theorem prover) to a many-sorted or one-sorted logic implemented in mainstream automated theorem provers. They devised a three-stage scheme where the last stage eliminates polymorphic types while adding the necessary "annotations" to preserve soundness, and the first two stages serve to protect certain terms so that they can keep their original types and unannotated form. Such protection allows to make use of provers' built-in theories and operations. This work generalizes the previous study by S. Lescuyer and J.-F. Couchot [65] onto arbitrary monomorphic types, e.g. array types. It was presented at FroCoS 2011 [22] (see also an extended version with full proofs [42]). These results are part of F. Bobot's PhD thesis [12].

6.4. Floating-Point and Numerical Programs

- T. Nguyen and C. Marché have worked on how to prove floating-point programs while taking into account architecture- and compiler-dependent features such as the use of the x87 stack in Intel microprocessors. This is done by analyzing the assembly code generated by the compiler [40], [28]
- S. Boldo and C. Marché published a survey article on the proofs of numerical C programs using both automatic provers and Coq [15].
- S. Boldo and T. Nguyen have worked on how to prove numerical programs on multiple architectures and compilers [17]. More precisely, it covers all the compiler choices about the use of extended registers, FMA, and reorganization of additions.
- S. Boldo and J.-M. Muller (CNRS, Arénaire, LIP, ÉNS Lyon) have worked on new floating-point algorithms for computing the exact and approximated errors of the FMA (fused multiply-and-add) [16].

- S. Boldo and G. Melquiond have developed in Coq a comprehensive formalization of floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties [23]. It provides a framework for developers to formally certify numerical applications.
- G. Melquiond, in collaboration with F. de Dinechin (Arénaire, LIP, ÉNS Lyon) and C. Lauter (Intel Hillsboro), has improved the methodology for formally proving floating-point mathematical functions when their correctness depends on relative errors [19].
- S. Boldo, J.-C. Filliâtre and G. Melquiond, in collaboration with F. Clément (Estime, INRIA Paris-Rocquencourt) and M. Mayero (University Paris 13) have finished a full formal proof of a program solving a partial differential equation (the wave equation) using a finite difference scheme [36]. This proof includes both the mathematical convergence proof (method error) [57], a tricky floating-point proof [56] and proofs of the absence of runtime errors.
- C. Lelay, under the supervision of S. Boldo and G. Melquiond, has worked on differentiability in Coq. The goal was to prove the existence of a solution to the wave equation thanks to D'Alembert's formula and to automatize the process as much as possible [44] [77].
- G. Melquiond, in collaboration with W. G. Nowak (Institute of Mathmatics, Austria) and P. Zimmermann (Caramel, INRIA Nancy-Lorraine), has designed new methods for computing guaranteed enclosures of the Masser-Gramain constant, a two-dimensional analogue of the Euler-Mascheroni constant [86].
- G. Melquiond, in collaboration with J-M. Muller (CNRS, Arénaire, LIP, ÉNS Lyon) and E. Martin-Dorel (Arénaire, LIP, ÉNS Lyon), has worked on weakening the assumptions floating-point error-free transformations rely on [39].

7. Contracts and Grants with Industry

7.1. Systematic: Hi-Lite

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

7.2. CEA-Airbus contract

Participants: Sylvain Conchon [contact], Évelyne Contejean, Claude Marché.

In conjunction with the INRIA funding of ADT Alt-Ergo, a specific support contract has started in Sep 09, between INRIA, CEA Saclay and Airbus France at Toulouse. This is to support our efforts for the maintainance and to feature updates of Alt-Ergo, for its use at Airbus software development and certification of avionics critical code.

7.3. 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.

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 years Digiteo project that started in September 2008. 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.

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. 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, Jean-Christophe Filliâtre, Xavier Urbain [contact].

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. U3CAT

Participants: Jean-Christophe Filliâtre, Claude Marché [contact], Guillaume Melquiond, Kalyan Krishnamani, 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 will end in 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).

8.2.2. INRIA ADT Alt-Ergo

Participants: Sylvain Conchon [contact], Evelyne Contejean, Claude Marché, Alain Mebsout, Mohamed Iguernelala.

The *ADT* (Action de Développement Technologique) Alt-Ergo is a 2-years project funded by INRIA, started in September 2009.

The goal is the maturation of the Alt-Ergo prover towards its use in an industrial context in particular for avionics. The expected outcomes of this ADT are the following:

- improving the efficiency of Alt-Ergo;
- fine tuning of Alt-Ergo for the SMT competition;
- generation of counter-examples;
- the qualification of Alt-Ergo for the norm DO-178B.

External Collaborators: Airbus France (Toulouse), Dassault Aviation (Saint-Cloud), team Typical (INRIA, École Polytechnique).

8.2.3. 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. 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. SCALP

Participants: Christine Paulin-Mohring [contact], David Baelde, Xavier Urbain.

This project is funded by ANR (program SESUR). http://scalp.gforge.inria.fr/

It started on january 2008 for 4 years; the coordinator is Yassine Lakhnech from VERIMAG.

The SCALP project (Security of Cryptographic Algorithms with Probabilities) aims at developing automated tools for the verification of cryptographic systems.

Partners: Verimag, INRIA Sophia-Antipolis(Everest then Marelle team), ENS Lyon, LRI, CNAM.

8.2.5. DECERT

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer.

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

8.3. European Initiatives

8.3.1. Collaborations in European Programs, except FP7

8.3.1.1. FoVeOOS

Participants: Claude Marché [contact], Romain Bardou, 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. Visits of International Scientists

- D. Ishii (National Institute of Informatics, Japan) visited the team for 8 months to work on applying program verification methods to hybrid systems.

8.4.2. Supervision of Post-docs and Internships

- S. Boldo supervised the 6-month post-doc intern of E. Makarov (from University of Vermont, USA) about numerical analysis proofs in higher dimensions.
- C. Marché supervised the post-doc intern of K. Krishnamani (from University of Trento, Italy) until August: predicate abstraction techniques for critical C programs ([38] http://proval.lri.fr/agen).
- S. Conchon and G. Melquiond supervise the post-doc intern of Cody Roux since May 2011: integration of the Gappa theorem prover in the Alt-Ergo SMT solver.
- S. Conchon supervises the post-doc intern of D. Cousineau since October 2011: interpretation of Alt-Ergo’s proof traces in the Coq proof assistant.
- C. Paulin supervised the internship of N. Gaspar (Universidade da Beira Interior, Portugal) from January to September 2011. He studied the formal proof of concurrent programs using a rely-guarantee approach.
- E. Contejean, together with V. Benzaken (LRI), supervise the internship of S. Yuan (Zhejiang University, China) from October 2011 to March 2012: Automated constraints verification for databases with SMT solvers.

8.4.3. 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 SummerSchool "Imagine the future in ICT".

9. Dissemination

9.1. Animation of the scientific community

9.1.1. Event organization

- At the VSTTE 2010 conference was organized a first and informal program verification competition (http://www.macs.hw.ac.uk/vstte10/Competition.html), as a prelude to more formal competitions in the future. It lasted for 2 hours, and participants had to submit solutions to five simple verification problems. There were 11 participants, and most of them solved only 2 problems.

  On behalf of the VSTTE 2012 conference (Philadelphia, USA, January 2012), A. Paskevich and J.-C. Filliâtre organized the first formal VSTTE program verification competition (https://sites.google.com/site/vstte2012/compet). This time it lasted for 48 hours, from November 8 to November 10. A set of five verification problems was proposed to the participants. The problems to solve were significantly harder than those of 2010. Each problem consisted of an algorithm given in pseudocode, together with a set of properties to be mechanically proved. A total of 29 teams (79 participants) sent solutions, which is considered an excellent success. These solutions are currently under examination. Results will be announced at the conference.

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. Marché co-edited with B. Beckert a special issue of Elsevier Lectures Notes in Computer Science devoted to selected papers of the conference FoVeOOS'10 [34].
- C. Paulin is member of the editorial board of the *Journal of Formalized Reasoning*.
- J.-C. Filliâtre edited a special issue of Software Tools for Technology Transfer devoted to selected papers of the workshop VSTTE 2009. This includes an introduction paper on deductive software verification [18].

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

- S. Conchon, program chair of the Journées Francophones des Langages Applicatifs (JFLA 2011), La Bresse, France, January 2011.
- S. Boldo, member of the program committee of JFLA 2011, and the Fourth International Workshop on Numerical Software Verification (NSV 2011, affiliated to CAV).
- D. Baelde, G. Melquiond, members of the program committee of JFLA 2012.
- É. Contejean is a member of program committees of the ACM SIGPLAN 2011 Workshop on Partial Evaluation and Program Manipulation (PEPM 2011, co-located with POPL, Austin, Texas), the International Workshop on Proof Search in Axiomatic Theories and Type Theories (PSATTT 20011, affiliated to CADE, Wroclaw, Poland).

- C. Marché, member of program committees of the 2nd International Conference on Formal Verification of Object-Oriented Software (FoVeOOS 2011, Turin, Italy), the 23rd International Conference on Automated Deduction (CADE 2011, Wroclaw, Poland), and the the first International Workshop on Intermediate Verification Languages (BOOGIE 2011, affiliated to CADE).
- C. Paulin, member of the program committee of the second and third conference on Interactive Theorem Proving (ITP 2011 & 2012), and the Fifth ACM SIGPLAN Workshop on Programming Languages meets Program Verification (PLPV 2011), affiliated to POPL.
- J.-C. Filliâtre, member of the program committee of the second conference on Interactive Theorem Proving (ITP 2011), the workshop "Analyze to Compile, Compile to Analyze" (ACCA 2011), and the conference Verified Software: Theories, Tools and Experiments (VSTTE 2012).

9.1.5. Participation to Thesis Committees

- C. Marché: president of PhD committee of Diego Caminha Barbosa de Oliveira (University Nancy 2, March 14th, 2011)
- C. Marché: reviewer, PhD of Beatriz Alarcón (University of Valencia, Spain, May 26th, 2011)
- C. Marché: reviewer, PhD of Mauricio Alba Castro (University of Valencia, Spain, Nov 25th, 2011)
- C. Marché: reviewer, PhD of Séverine Maingaud (University Paris 7, Dec 13th, 2011)
- C. Paulin: examinator, PhD of Mathias Krieger (University Paris-Sud 11, Dec 9th, 2011)

9.1.6. Invited Presentations

- S. Boldo, "Contours de la communauté", invited talk at the *4es Rencontres Arithmétique de l'Informatique Mathématique* (RAIM'11) in Perpignan. (Collected data about the outline of the computer arithmetic community in France: sites, themes, fundings...).
- J.-C. Filliâtre, "Memo Tables", invited at the IFIP Working Group 2.8 *Functional Programming* (Marble Falls, Texas, USA, March 7–11, 2011).
- P. Herms, "Certification of a Verification Condition Generator in Coq", seminar of the Gallium-Moscova teams, Rocquencourt, June 20th.
- C. Marché, "Verifying Behavioral Specifications of Programs: the Why Approach", seminar of the ELP team, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, March 25th.
- T. Nguyen, "Hardware-independent proofs of numerical programs ", seminar of the Arenaire team, Lyon, January 20th.
- C. Paulin, "About Inductive-Recursive Definitions in Coq", invited speaker at the workshop on Proofs and Programs, Gothenburg, Sweden, Oct. 22th.

9.2. Interaction with the scientific community

9.2.1. Prizes and distinctions

- S. Conchon, E. Contejean and M. Iguernelala got the award 2011 of the European Association for Theoretical Computer Science http://www.eatcs.org/index.php/best-etaps-paper for the best theoretical paper of all ETAPS Conferences [24].
- C. Paulin received an honorary doctorate from the University of Gothenburg in Sweden on October 21, 2011.

9.2.2. Collective responsibilities within INRIA

- S. Boldo, elected member of the Inria Evaluation Committee.

- 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, member of the MECSI group for networking about computer science popularization inside INRIA.
- C. Paulin participates to the board of INRIA Saclay - Île-de-France Comité des Projets (assembly of Team leaders).
- C. Paulin is a member of the “Commission Scientifique” (in charge of selecting PhD students, post-doc, invited researchers funded by INRIA Saclay - Île-de-France).

9.2.3. Collective responsibilities outside INRIA

- E. Contejean and C. Marché, nominated members of the *“conseil du laboratoire”* of LRI since April 2010.
- C. Marché, French National Coordinator for the COST action “Formal Verification of Object-Oriented Programs” (2008-2012).
- 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, member of the selection committee of the “DIM Logiciels et Systèmes Complexes”, providing grants to research projects, funded by *Île-de-France* regional council and Digiteo cluster, http://www.dimlsc.fr/.
- G. Melquiond, elected officer of the IEEE-1788 standardization committee on interval arithmetic since 2008.
- G. Melquiond, C. Paulin, members of the *“commission consultative de spécialistes de l’université”*, Section 27, University Paris-Sud since April 2010.
- G. Melquiond is an examiner for the computer science entrance exam to École Normale Supérieure since 2010.
- C. Paulin, director of the Graduate school in Computer Science at University Paris Sud http://edips.lri.fr/.
- C. Paulin is deputy director of the LRI.
- C. Paulin was the president of an hiring committee for a professor position at University Paris-Sud. She participated to two hiring commitees (professor position at ENSEEIHT Toulouse and assistant professor for the PPS laboratory at University Paris Diderot)
- C. Paulin participated to the review panels for the German Excellence Initiative proposals for Graduate Schools in informatics.
- 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.
- X. Urbain, hiring committee for an assistant professor position at École Centrale, Paris (Spring 2011).
- X. Urbain is an elected member of the board (*“conseil d’administration”*) of École Nationale Supérieure d’Informatique pour l’Industrie et l’Entreprise (ENSIIE). Since July he is head of the teaching departement of ENSIIE.

9.3. Industrial Dissemination

- 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.
- As part of the qualification process of Alt-Ergo with Airbus industry (DO-178B), the technical documents (functional specifications and benchmark suite) have been accepted by Airbus. These documents will be submitted by Airbus to the certification authorities in 2012.
- S. Conchon has started a collaboration with S. Krstic and A. Goel (Intel Strategic Cad Labs in Hillsboro, OR, USA) that aims in the development of an SMT-based model checker. With A. Mebsout and F. Zaidi (ForTesSe, LRI), they implement the Cubicle model checker which uses the Alt-Ergo theorem prover to discharge its proof obligations.
- The Adacore company (Paris) implements a new tool GnatProve which aims at formal verification of Ada programs. They translate annotated Ada code into the *Why*3 intermediate language and then use the *Why*3 system to generate proof obligations and discharge them with available back-end provers.
- J.-C. Filliâtre and C. Marché have started a collaboration with D. Mentré at Mitsubishi Electric R&D Centre Europe (Rennes), about the use of the *Why*3 environment and its back-end provers as an alternative to the built-in prover of Atelier B.

9.4. Popularization

Since April 2008, S. Boldo is member of the editorial committee of the popular science web site)i(: http://interstices.info/.

Since July 2009, S. Boldo is elected member of the board of the Animath association that promotes mathematics among young people.

S. Boldo, in collaboration with T. Viéville (INRIA Nancy Grand-Est) wrote two chapters of the book “Introduction à la science informatique”, edited by G. Dowek [32], [33]. This book aims at helping the secondary school teachers for the incoming computer science teaching.

9.5. Teaching

Licence (DUT): “Programmation Java” (L2), “Projet professionnel et Personnel” (L1), “Architecture” (L1), “Bases de données” (L2), A. Tafat (64h, “moniteur” position), Université Paris-Sud (IUT d’Orsay), France

Licence (DUT): “Systèmes d’exploitation” (L1), “Architecture des ordinateurs” (L1), M. Iguernelala (64h, “moniteur” position), Université Paris-Sud (IUT d’Orsay), France

Licence (DUT): “Systèmes d’exploitation” (L1 and L2), “Réseaux” (L2), A. Paskevich (156h), Université Paris-Sud (IUT d’Orsay), France

Licence : “Mathématiques pour l’Informatique” (L2), C. Paulin (50h), D. Baelde (20h), Université Paris-Sud, France

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

Licence: “Langages de programmation et compilation” (L3), J.-C. Filliâtre (24h), École Normale Supérieure, France

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

Licence: “Programmation fonctionnelle” (L3), S. Conchon (50h), Université Paris-Sud, France

Licence: “Programmation fonctionnelle” (L3), “Projet de programmation” (L3), F. Bobot (64h, “moniteur” position), Université Paris-Sud, France

Master: “Compilation” (M1), C. Paulin (50h), D. Baelde (28h), Université Paris-Sud, France

Master: “Projet de compilation” (M1), R. .Bardou (64h, “moniteur” position), Université Paris-Sud, France

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/: “Proof assistants” (M2), G. Melquiond (9h), C. Paulin (6h), Université Paris 7, France

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/: “Automated deduction” (M2), É. Contejean (12h), X. Urbain (12h), Université Paris 7, France

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/: “Foundations of proof system” (M2), S. Boldo (2h), Université Paris 7, France

8th LASER Summer School on Software Engineering http://laser.inf.ethz.ch/2011/: “Tools for Practical Software Verification”, C. Paulin (4h)

Supervision of internships

S. Boldo and G. Melquiond supervised the internship of C. Lelay about differentiability in Coq [44] (Master Logique Mathématique et Fondements de l’Informatique, Univ. Paris Diderot).

PhD & HdR:

HDR: J.-C. Filliâtre, Deductive Program Verification [13], Université Paris-Sud, Dec. 2nd 2011

PhD: S. Lescuyer, Formalizing and Implementing a Reflexive Tactic for Automated Deduction in Coq [14], Université Paris-Sud, Jan. 4th 2011, S. Conchon and É. Contejean

PhD: R. Bardou, Verification of Pointer Programs Using Regions and Permissions [11], Université Paris-Sud, Oct. 14th 2011, C. Marché

PhD: F. Bobot, Logique de séparation et vérification déductive [12], Université Paris-Sud, Dec. 12 2011, J.-C. Filliâtre

PhD in progress: T. Nguyen, Formal Proof of Numerical Programs with respect to Architecture and Compiler, since February 2009, 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)

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

PhD in progress: Alain 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: A. J. Compaore, Rewriting Techniques for (Space and Time) Simulation of Biological Processes, since November 2007, defence in Feb. 2012, X. Urbain and P. Le Gall (ECP & Université Évry).

PhD in progress: Z. Bouzid, Models and Algorithms for Emerging Systems, since October 2009, X. Urbain and S. Tixeuil, M. Gradinariu Potop-Butucaru (Université Paris 6).

PhD stopped: W. Urribarrí, Towards certified libraries, from Nov. 2006 to Sep. 2011. W. Urribarrí is now an engineer “development of secure software” at ClearSy.

10. Bibliography

Major publications by the team in recent years

[1] 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.

[2] S. BOLDO, J.-C. FILLIÂTRE. *Formal Verification of Floating-Point Programs*, in "18th IEEE International Symposium on Computer Arithmetic", Montpellier, France, June 2007, p. 187-194, http://www.lri.fr/~filliatr/ftp/publis/caduceus-floats.pdf.

[3] S. CONCHON, É. CONTEJEAN, J. KANIG, S. LESCUYER. *CC(X): Semantical Combination of Congruence Closure with Solvable Theories*, in "Post-proceedings of the 5th International Workshop on Satisfiability Modulo Theories (SMT 2007)", Electronic Notes in Computer Science, Elsevier Science Publishers, 2008, vol. 198-2, p. 51–69.

[4] É. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS, X. URBAIN. *Certification of automated termination proofs*, in "6th International Symposium on Frontiers of Combining Systems (FroCos 07)", Liverpool,UK, B. KONEV, F. WOLTER (editors), Lecture Notes in Artificial Intelligence, Springer, September 2007, vol. 4720, p. 148–162.

[5] É. CONTEJEAN, C. MARCHÉ, A. P. TOMÁS, X. URBAIN. *Mechanically proving termination using polynomial interpretations*, in "Journal of Automated Reasoning", 2005, vol. 34, n^o 4, p. 325–363, http://dx.doi.org/10.1007/s10817-005-9022-x.

[6] J.-C. FILLIÂTRE. *Verification of Non-Functional Programs using Interpretations in Type Theory*, in "Journal of Functional Programming", July 2003, vol. 13, n^o 4, p. 709–745, http://www.lri.fr/~filliatr/ftp/publis/jphd.pdf.

[7] J.-C. FILLIÂTRE, C. MARCHÉ. *Multi-Prover Verification of C Programs*, in "6th International Conference on Formal Engineering Methods", Seattle, WA, USA, J. DAVIES, W. SCHULTE, M. BARNETT (editors), Lecture Notes in Computer Science, Springer, November 2004, vol. 3308, p. 15–29, http://www.lri.fr/~filliatr/ftp/publis/caduceus.ps.gz.

[8] T. HUBERT, C. MARCHÉ. *A case study of C source code verification: the Schorr-Waite algorithm*, in "3rd IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)", Koblenz, Germany, B. K. AICHERNIG, B. BECKERT (editors), IEEE Comp. Soc. Press, September 2005, http://www.lri.fr/~marche/hubert05sefm.ps.

[9] S. KRSTIĆ, S. CONCHON. *Canonization for disjoint unions of theories*, in "Information and Computation", May 2005, vol. 199, n^o 1-2, p. 87–106.

[10] C. MARCHÉ, C. PAULIN-MOHRING, X. URBAIN. *The* KRAKATOA *Tool for Certification of* JAVA/JAVACARD *Programs annotated in* JML, in "Journal of Logic and Algebraic Programming", 2004, vol. 58, n^o 1–2, p. 89–106, http://krakatoa.lri.fr.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] R. BARDOU. *Verification of Pointer Programs Using Regions and Permissions*, Université Paris-Sud, October 2011, http://romain.bardou.fr/thesis/bardou11phd.pdf.

[12] F. BOBOT. *Logique de séparation et vérification déductive*, Université Paris-Sud, December 2011.

[13] J.-C. FILLIÂTRE. *Deductive Program Verification*, Université Paris-Sud, December 2011, Thèse d'habilitation.

[14] S. LESCUYER. *Formalisation et développement d'une tactique réflexive pour la démonstration automatique en Coq*, Université Paris-Sud, January 2011.

Articles in International Peer-Reviewed Journal

[15] S. BOLDO, C. MARCHÉ. *Formal verification of numerical programs: from C annotated programs to mechanical proofs*, in "Mathematics in Computer Science", 2011.

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

[17] 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.

[18] 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://dx.doi.org/10.1007/s10009-011-0211-0.

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

International Conferences with Proceedings

[20] D. BAELDE, R. BEAUXIS, S. MIMRAM. *Liquidsoap: A High-Level Programming Language for Multimedia Streaming*, in "37th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'11)", Nový Smokovec, Slovakia, I. CERNÁ, T. GYIMÓTHY, J. HROMKOVIC, K. G. JEFFERY, R. KRÁLOVIC, M. VUKOLIC, S. WOLF (editors), Lecture Notes in Computer Science, Springer, January 2011, vol. 6543.

[21] 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, http://proval.lri.fr/submissions/boogie11.pdf.

[22] 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.

[23] 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://www.lri.fr/~melquion/doc/11-arith20-article.pdf.

[24] *Best Paper*
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.

[25] 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.

[26] C. Dross, J.-C. Filliâtre, Y. Moy. *Correct Code Containing Containers*, in "5th International Conference on Tests and Proofs (TAP'11)", Zurich, June 2011.

[27] J.-C. Filliâtre, K. Kalyanasundaram. *Functory: A Distributed Computing Library for Objective Caml*, in "Trends in Functional Programming", Madrid, Spain, May 2011.

[28] 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.

[29] A. Tafat, S. Boulmé, C. Marché. *A Refinement Methodology for Object-Oriented Programs*, in "Formal Verification of Object-Oriented Software, Revised Selected Papers Presented at the International Conference, FoVeOOS 2010", B. Beckert, C. Marché (editors), Lecture Notes in Computer Science, Springer, January 2011, vol. 6528, p. 153–167.

National Conferences with Proceeding

[30] R. Bardou, C. Marché. *Perle de preuve: les tableaux creux*, in "Vingt-deuxièmes Journées Francophones des Langages Applicatifs", La Bresse, France, S. Conchon (editor), INRIA, January 2011.

[31] 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.

Scientific Books (or Scientific Book chapters)

[32] S. Boldo, T. Viéville. *Représentation numérique de l'information*, in "Introduction à la science informatique", G. Dowek (editor), Repères pour agir, CRDP Académie de Paris, July 2011, p. 23–72, http://crdp.ac-paris.fr/Introduction-a-la-science.

[33] S. Boldo, T. Viéville. *Structuration et contrôle de l'information*, in "Introduction à la science informatique", G. Dowek (editor), Repères pour agir, CRDP Académie de Paris, July 2011, p. 281–308, http://crdp.ac-paris.fr/Introduction-a-la-science.

Books or Proceedings Editing

[34] B. BECKERT, C. MARCHÉ (editors). *Formal Verification of Object-Oriented Software, Revised Selected Papers Presented at the International Conference, FoVeOOS 2010*, Lecture Notes in Computer Science, Springer, January 2011, vol. 6528.

Research Reports

[35] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, A. PASKEVICH. *The Why3 platform*, version 0.64, LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, February 2011.

[36] S. BOLDO, F. CLEMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Wave Equation Numerical Resolution: Mathematics and Program*, INRIA, December 2011, n^{o} RR-7826, http://hal.inria.fr/hal-00649240/en/.

[37] P. HERMS, C. MARCHÉ, B. MONATE. *A Certified Multi-prover Verification Condition Generator*, INRIA, 2011, n^{o} 7793, http://hal.inria.fr/hal-00639977/en/.

[38] 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/.

[39] É. MARTIN-DOREL, G. MELQUIOND, J.-M. MULLER. *Some issues related to double roundings*, 2011, http://hal-ens-lyon.archives-ouvertes.fr/ensl-00644408/en/.

[40] T. M. T. NGUYEN, C. MARCHÉ. *Proving Floating-Point Numerical Programs by Analysis of their Assembly Code*, INRIA, 2011, n^{o} 7655, http://hal.inria.fr/inria-00602266/en/.

[41] A. TAFAT, C. MARCHÉ. *Binary Heaps Formally Verified in Why3*, INRIA, October 2011, n^{o} 7780, http://hal.inria.fr/inria-00636083/en/.

Other Publications

[42] F. BOBOT, A. PASKEVICH. *Expressing Polymorphic Types in a Many-Sorted Language*, 2011.

[43] N. GASPAR. *Mechanized Semantics into Concurrent Program verification*, September 2011, http://www.lri.fr/~gaspar/rgcoq.html.

[44] C. LELAY. *Étude de la différentiabilité et de l'intégrabilité en Coq : Application à la formule de d'Alembert pour l'équation des ondes*, Université Paris 7, 2011.

References in notes

[45] *The MAUDE System*.

[46] J. ANDRONICK. *Modélisation et vérification formelles de systèmes embarqués dans les cartes à microprocessur. Plateforme Java Card et Système d'exploitation*, Université Paris-Sud, March 2006, http://ssrg.nicta.com.au/publications/papers/Andronick:phd.abstract?bib=login.

[47] J. ANDRONICK, B. CHETALI, C. PAULIN-MOHRING. *Formal Verification of Security Properties of Smart Card Embedded Source Code*, in "International Symposium of Formal Methods Europe (FM'05)", Newcastle,UK, J. FITZGERALD, I. J. HAYES, A. TARLECKI (editors), Lecture Notes in Computer Science, Springer, July 2005, vol. 3582, http://www.springerlink.com/content/eulj9pbgm2875cer/.

[48] T. ARTS, J. GIESL. *Termination of term rewriting using dependency pairs*, in "Theoretical Computer Science", 2000, vol. 236, p. 133–178.

[49] 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/.

[50] 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.

[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. *Verification of the Interface of a Small Proof System in Coq*, in "Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers", E. GIMÉNEZ, C. PAULIN-MOHRING (editors), Lecture Notes in Computer Science, Springer, 1998, vol. 1512, p. 28-45.

[53] 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.

[54] 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.

[55] S. BOLDO. *Pitfalls of a full floating-point proof: example on the formal proof of the Veltkamp/Dekker algorithms*, in "Third International Joint Conference on Automated Reasoning", Seattle, USA, U. FURBACH, N. SHANKAR (editors), Lecture Notes in Computer Science, Springer, August 2006, vol. 4130, p. 52-66, http://www.springerlink.com/content/524v5246177t0877/.

[56] 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.

[57] 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/en/.

[58] R. BORNAT. *Proving Pointer Programs in Hoare Logic*, in "Mathematics of Program Construction", 2000, p. 102–126.

[59] V. CHAUDHARY. *The Krakatoa tool for certification of Java/JavaCard programs annotated in JML : A Case Study*, IIT internship report, July 2004.

[60] S. CONCHON, S. KRSTIĆ. *Strategies for Combining Decision Procedures*, in "Theoretical Computer Science", 2006, vol. 354, n^{o} 2, p. 187–210.

[61] É. CONTEJEAN, P. CORBINEAU. *Reflecting Proofs in First-Order Logic with Equality*, in "20th International Conference on Automated Deduction (CADE-20)", Tallinn, Estonia, R. NIEUWENHUIS (editor), Lecture Notes in Artificial Intelligence, Springer, July 2005, vol. 3632, p. 7–22.

[62] É. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS, X. URBAIN. *Certification of automated termination proofs*, CEDRIC, May 2007, n^{o} 1185.

[63] É. CONTEJEAN, C. MARCHÉ, A. P. TOMÁS, X. URBAIN. *Mechanically proving termination using polynomial interpretations*, in "Journal of Automated Reasoning", 2005, vol. 34, n^{o} 4, p. 325–363, http://dx.doi.org/10.1007/s10817-005-9022-x.

[64] J.-F. COUCHOT, A. GIORGETTI, N. STOULS. *Graph-based Reduction of Program Verification Conditions*, INRIA Saclay – Île-de-France, October 2008, n^{o} 6702, http://hal.inria.fr/inria-00339847/en/.

[65] J.-F. COUCHOT, S. LESCUYER. *Handling Polymorphism in Automated Deduction*, in "21th International Conference on Automated Deduction (CADE-21)", Bremen, Germany, LNCS (LNAI), July 2007, vol. 4603, p. 263–278.

[66] 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/.

[67] F. DURÁN, S. LUCAS, J. MESEGUER, C. MARCHÉ, X. URBAIN. *Proving Termination of Membership Equational Programs*, in "ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation", Verona, Italy, ACM Press, August 2004.

[68] J.-C. FILLIÂTRE. *Formal Proof of a Program: Find*, in "Science of Computer Programming", 2006, vol. 64, p. 332–240, http://www.lri.fr/~filliatr/ftp/publis/find.pdf.

[69] J.-C. FILLIÂTRE. *Verifying Two Lines of C with Why3: an Exercise in Program Verification*, in "Verified Software: Theories, Tools and Experiments (VSTTE)", Philadelphia, USA, January 2012, http://why3.lri.fr/queens/queens.pdf.

[70] J.-C. FILLIÂTRE, S. OWRE, H. RUESS, N. SHANKAR. *ICS: Integrated Canonization and Solving (Tool presentation)*, in "Proceedings of CAV'2001", G. BERRY, H. COMON, A. FINKEL (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2102, p. 246–249.

[71] 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.

[72] B. GRAMLICH. *On Proving Termination by Innermost Termination*, in "7th International Conference on Rewriting Techniques and Applications", New Brunswick, NJ, USA, H. GANZINGER (editor), Lecture Notes in Computer Science, Springer, July 1996, vol. 1103, p. 93–107.

[73] T. HUBERT. *Analyse Statique et preuve de Programmes Industriels Critiques*, Université Paris-Sud, June 2008, http://www.lri.fr/~marche/hubert08these.pdf.

[74] T. HUBERT, C. MARCHÉ. *Separation Analysis for Deductive Verification*, in "Heap Analysis and Verification (HAV'07)", Braga, Portugal, March 2007, p. 81–93, http://www.lri.fr/~marche/hubert07hav.pdf.

[75] B. JACOBS, C. MARCHÉ, N. RAUCH. *Formal Verification of a Commercial Smart Card Applet with Multiple Tools*, in "Algebraic Methodology and Software Technology", Stirling, UK, Lecture Notes in Computer Science, Springer, July 2004, vol. 3116.

[76] K. R. M. LEINO, M. MOSKAL. *VACID-0: Verification of Ample Correctness of Invariants of Data-structures, Edition 0*, in "Proceedings of Tools and Experiments Workshop at VSTTE", 2010.

[77] 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/.

[78] X. LEROY. *Formal certification of a compiler back-end, or: programming a compiler with a proof assistant*, in "Conference Record of the 33rd Symposium on Principles of Programming Languages", Charleston, South Carolina, ACM Press, January 2006.

[79] S. LESCUYER. *Codage de la logique du premier ordre polymorphe multi-sortée dans la logique sans sortes*, Master Parisien de Recherche en Informatique, 2006.

[80] P. LETOUZEY. *A New Extraction for Coq*, in "TYPES 2002", H. GEUVERS, F. WIEDIJK (editors), Lecture Notes in Computer Science, Springer, 2003, vol. 2646.

[81] P. LETOUZEY. *Programmation fonctionnelle certifiée: l'extraction de programmes dans l'assistant Coq*, Université Paris-Sud, July 2004.

[82] C. MARCHÉ, C. PAULIN-MOHRING. *Reasoning about Java Programs with Aliasing and Frame Conditions*, in "18th International Conference on Theorem Proving in Higher Order Logics", J. HURD, T. MELHAM (editors), Lecture Notes in Computer Science, Springer, August 2005, vol. 3603, p. 179–194, http://www.lri.fr/~marche/marche05tphols.ps.

[83] C. MARCHÉ, N. ROUSSET. *Verification of Java Card Applets Behavior with respect to Transactions and Card Tears*, in "4th IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)", Pune, India, D. V. HUNG, P. PANDYA (editors), IEEE Comp. Soc. Press, September 2006.

[84] C. MARCHÉ, X. URBAIN. *Modular and Incremental Proofs of AC-Termination*, in "Journal of Symbolic Computation", 2004, vol. 38, p. 873–897, http://authors.elsevier.com/sd/article/S074771710400029X.

[85] C. MARCHÉ. *Preuves mécanisées de Propriétés de Programmes*, Université Paris 11, December 2005, Thèse d'habilitation.

[86] G. MELQUIOND, W. G. NOWAK, P. ZIMMERMANN. *Numerical Approximation of the Masser-Gramain Constant to Four Decimal Digits: delta=1.819...*, in "Mathematics of Computation", 2012, http://hal.inria.fr/hal-00644166/en/.

[87] 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/.

[88] Y. MOY. *Automatic Modular Static Safety Checking for C Programs*, Université Paris-Sud, January 2009, http://www.lri.fr/~marche/moy09phd.pdf.

[89] E. OHLEBUSCH, C. CLAVES, C. MARCHÉ. *TALP: A Tool for the Termination Analysis of Logic Programs*, in "11th International Conference on Rewriting Techniques and Applications", Norwich, UK, L. BACHMAIR (editor), Lecture Notes in Computer Science, Springer, July 2000, vol. 1833, p. 270–273.

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

[91] M. SOZEAU. *Program-ing Finger Trees in Coq*, in "12th ACM SIGPLAN International Conference on Functional Programming, ICFP 2007", Freiburg, Germany, R. HINZE, N. RAMSEY (editors), ACM Press, 2007, p. 13–24.

[92] M. SOZEAU. *Un environnement pour la programmation avec types dépendants*, Université Paris-Sud, December 2008.

[93] D. STEVENSON. *A proposed standard for binary floating point arithmetic*, in "IEEE Computer", 1981, vol. 14, n^{o} 3, p. 51-62.

[94] X. URBAIN. *Approche incrémentale des preuves automatiques de terminaison*, Université Paris-Sud, Orsay, France, October 2001, http://www.lri.fr/~urbain/textes/these.ps.gz.