INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Proval

Proof of programs

Saclay - Île-de-France

Theme : Programs, Verification and Proofs

Table of contents

The Proval project-team is a research team common to INRIA - Saclay Île-de-France, CNRS and Université Paris-Sud 11. Researchers are also members of the LRI (Laboratoire de Recherche en Informatique, UMR 8623).

1. Team

Research Scientist

Claude Marché [Team Vice-Leader, DR INRIA, HdR]
Sylvie Boldo [CR INRIA]
Évelyne Contejean [CR CNRS]
Jean-Christophe Filliâtre [CR CNRS]
Guillaume Melquiond [CR INRIA]

Faculty Member

Christine Paulin-Mohring [Team Leader, Professor University Paris-Sud 11, HdR]
Marc Pouzet [Professor University Paris-Sud 11, member of IUF, HdR]
Sylvain Conchon [Assistant Professor Université Paris-Sud 11, delegation INRIA since Sep. 09]
Louis Mandel [Assistant Professor Université Paris-Sud 11]
Andrei Paskevich [Assistant Professor Université Paris-Sud 11, since Sep. 09]
Xavier Urbain [Assistant Professor ENSIIE, Delegation CNRS since Sep. 09]

Technical Staff

Alain Mebsout [Engineer INRIA, since Oct. 09]

PhD Student

Cédric Auger [University Paris-Sud 11]
Romain Bardou [University Paris-Sud 11]
Nicolas Bertaux [CORDI INRIA]
François Bobot [ÉNS Cachan until Aug. 09, then University Paris-Sud 11]
Léonard Gérard [ÉNS Lyon until Aug. 09, then University Paris-Sud 11]
Paolo Herms [CEA grant, since Oct. 09]
Mohamed Iguernelala [University Paris-Sud 11, since Sep. 09]
Johannes Kanig [CORDI INRIA]
Stéphane Lescuyer [INRIA, on leave from X-Mines]
Yannick Moy [CIFRE France Télécom, Lannion, until Jan. 09]
Thi-Minh-Tuyen Nguyen [INRIA Digiteo grant, since Feb. 09]
Florence Plateau [University Paris-Sud 11]
Asma Tafat Bouzid [University Paris-Sud 11, since Sep. 09]
Wendi Urribarrí [France-Venezuela scholarship]

Post-Doctoral Fellow

Ali Ayad [Post-doc CNRS, until Jan. 09]
Krishnamani Kalyanasundaram [Post-doc INRIA, since Sep. 09]

Administrative Assistant

Régine Bricquet [TR INRIA]

Other

Arthur Milchior [ENS Paris, undergraduate internship, June to August 09]
Julien Robert [University Paris-Sud 11, Jane Street Summer Project, June to August 09]
Guillaume Von Tokarski [University Paris-Sud 11, Jane Street Summer Project, June to August 09]

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 (*Caduceus*) 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.

Marc Pouzet joined the team as a full professor in September 2005, opening a research activity on synchronous systems. The goal is to propose high-level languages for the development of critical embedded systems with high temporal constraints.

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

- Higher-order functional languages,
- Proof of imperative and object-oriented programs,
- Automated deduction for program proof,
- Synchronous Programming.

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

2.2. Highlights of the year

As part of the CerPAN and FOST projects, S. Boldo studied a real-life program computing the discretization of the spread of acoustic waves on a rope. She developed an appropriate technique to bound the rounding error due to the floating-point approximations of the computed values. As the naive approach fails, she provided a precise but complex analytical expression of the rounding error that gives a very good bound on the rounding error of the program. An article describing this application and this technique has been published in the very selective conference ICALP (36th International Colloquium on Automata, Languages and Programming) [19].

3. Scientific Foundations

3.1. Higher-Order Functional Languages

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 order to reach this goal, we have explored different directions.

3.1.1. Developing correct functional programs

Dependent types provide a powerful language for building programs that are correct by construction. In the language underlying the *Coq* proof assistant, it is possible to introduce the type of even numbers, or the type of sorted arrays of size n, or the type of correct compilers.

However for the type-checking to remain decidable, the program itself needs to contain many extra informations which are only used for correctness. This leads to two problems: the first one is how to write such programs in a natural way (we want to mainly describe the algorithm and let proof strategies find most of the correctness part); the second one is how to compute efficiently with these programs.

The first problem has been addressed by M. Sozeau who proposes an extension of the *Coq* input language for building programs [105]. The solution is similar to the mechanism of subset types and Type Checking Conditions in PVS but a *Coq* proof term is built and is checked by the *Coq* kernel. We are also working on extending the source language of *Coq* to support common abstractions of high-level functional languages like Haskell. M. Sozeau developed in collaboration with N. Oury a system of Type Classes in *Coq* [104] which gives overloading in programs and proofs and facilitates the development of generic tactics.

The second problem has been addressed in the *Coq* proof assistant by providing an extraction mechanism from *Coq* terms to purely functional programs (in Ocaml or Haskell) which are correct by construction. During his PhD thesis, P. Letouzey designed and implemented a new extraction mechanism for the *Coq* system [93], [94], much more powerful than the old version and together with J.-C. Filliâtre used it to verify Ocaml finite sets libraries based on balanced trees [6].

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 [91].

3.1.2. Using Type theory to model complex programs

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.

3.1.2.1. Randomized algorithms

C. Paulin in collaboration with Ph. Audebaud from ENS Lyon, proposed a method for modeling probabilistic programs in *Coq*. 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. It contains an axiomatisation of the real interval $[0, 1]$, a definition of distributions and general rules for approximating the probability that a program satisfies a given property.

3.1.2.2. Floating-point programs

Many industrial programs (weather forecasts, plane trajectories, simulations...) use floating-point computations, typically double precision floating-point numbers [106]. 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 [57].

We now have a methodology to perform formal verification of floating-point C programs. It extends the *Why* platform with new annotations specific to floating-point arithmetic. This technique is very flexible as both non-specialists and floating-point experts will be able to express the properties they assume the program to fulfil, directly on the source code. The generated VCs are for example "there is no overflow" or "the final error is less than..." [1].

3.1.2.3. 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.

We have also started a project for the modeling and proof of the correctness of a compiler for the Lustre synchronous language. Our goal is to show the feasability of the certification using formal proofs of the compiler used in the new version of Scade developed by Esterel Technologies.

3.2. Proof of Imperative and Object-Oriented programs

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Johannes Kanig, Claude Marché, Yannick Moy, Christine Paulin-Mohring, Wendi Urribarrí.

A foundation step of the project is the PhD thesis of Jean-Christophe Filliâtre [5] 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* [85].

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 (such as PVS, Isabelle/HOL, etc.) or with automatic provers (such as Simplify, SMT solvers, etc.). This multi-prover architecture is a powerful feature of *Why*: it spreads this technology well beyond the *Coq* community.

Since 2002, we tackle programs written in "real" 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, in collaboration with Gemalto company and Dassault Aviation company, and started the development of a tool called *Caduceus* [7].

3.2.1. The Why platform

We develop a platform combining several of our own tools and other ones. The tool playing the central role in our platform is *Why*, implementing the proof of programs approach proposed by Jean-Christophe Filliâtre [5]. The programs handled by *Why* are written in a specific language, they are annotated with pre and post

conditions (similar to classical Hoare's logic). *Why* generates VCs, the validity of which ensures correctness of the program with respect to the original specification. In *Why*, these VCs are first-order formula that can be translated into the syntax of different provers: interactive higher-order provers like *Coq*, PVS, HOL-light or Mizar or automatic provers such as Simplify, *Alt-Ergo*, haRVey or SMT provers (Yices, Z3, CVC3). This multi-prover architecture is clearly a strong advantage of the tool. *Why* is a tool which is regularly evolving. We integrate aspects which are not necessarily in the theory but which are needed for practical applications.

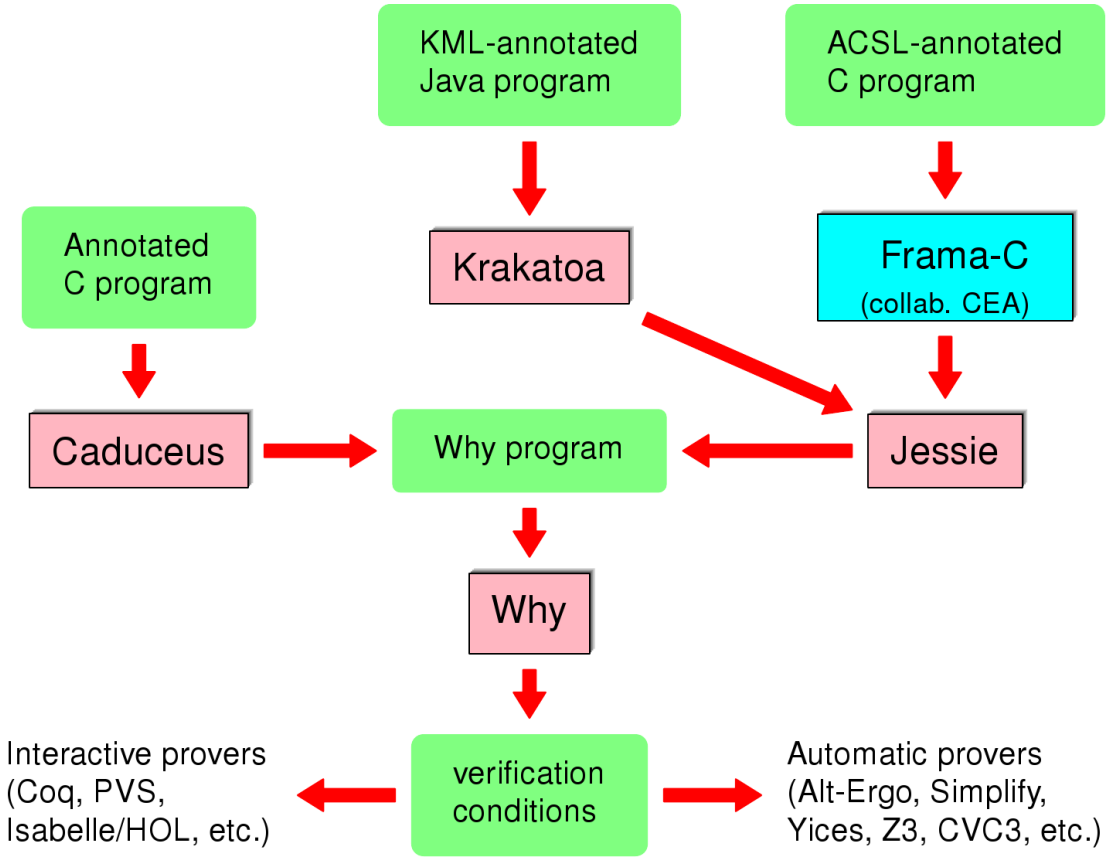

Figure 1. Overview of the architecture of Why and Frama-C

We develop *Why* front-ends for dealing with real C or Java source code. Our approach is based on a translation from source code into an equivalent program written in *Why*, leading to the architecture shown in Figure 1. 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* tool 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, *Why* 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 [98]. The *Krakatoa* and the *Caduceus* 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. 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].

3.2.2. The Frama-C Platform

We are developing, in collaboration with CEA-List, a platform called *Frama-C* for static analysis of C programs (http://www.frama-c.cea.fr/). This platform has an open architecture, structured as plugins around a shared kernel. The kernel reuse the CIL software from Berkeley University (USA) for parsing and typing C source code and annotations. Plugins include abstract interpretation techniques (analyses of values, aliasing, effects), Verification Condition Generators, code slicing. In particular, a part of our *Why* platform called *Jessie* is available as a *Frama-C* plugin.

This platform is under development, as part of the ANR CAT project, and is distributed under open-source licence.

3.2.3. 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 [89] 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 datas, 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 [62] 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 [46], [96].

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

3.3. Automated deduction

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer, Claude Marché, 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* [110], and further generalizations [97]. These criteria and methods have been implemented into the CiME2 rewrite toolbox [4]. 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 [101] and then to membership equational programs [84], 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 [86] 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], [76].

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. S. Lescuyer worked on this topic during his master thesis [92].

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 2009, *Alt-Ergo* is still the only existing prover dealing with parametric polymorphism.

Alt-Ergo is based on $CC(X)$, 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.

Alt-Ergo is safe and its architecture 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 (6500 lines).

The current experimentations are very promising with respect to speed and to the number of proof obligations automatically solved.

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 [50].

3.3.3.1. Coccinelle and CiME's traces

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* [77].

É. Contejean and the Cédric participants of the A3PAT project, Pierre Courtieu, Olivier Pons (CNAM), Julien Forest, and Xavier Urbain (ENSIIE) are currently developing a new version of the CiME tool 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 [80][3]. 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 [47], the main modularity theorem for termination presented in the thesis of Urbain [110] as well as innermost termination, dependency pairs for it and its equivalence with standard termination in some specific cases [87].

The main improvement over the previous approach [77] 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.

Since 2007, the termination competition has a new category for certified termination proofs. CiME\Coccinelle ranked the second place among the three participants in this category.

3.4. Synchronous Programming

Participants: Cédric Auger, Nicolas Bertaux, Léonard Gérard, Louis Mandel, Florence Plateau, Marc Pouzet.

The goal is to propose high-level languages for the development of critical embedded systems with both high temporal requirements and safety [51], [88], [52], [59]. Our research activities concern the extension of synchronous languages with richer abstraction mechanisms (e.g., higher-order, functionality, dedicated type systems such as the clock calculus), the ability to describe heterogeneous systems (e.g., data-flow and control-flow, discrete and continous) or to account for resources through dedicated type-systems.

These research activities are experimented inside two programming languages, Lucid Synchrone and ReactiveML.

Lucid Synchrone is a data-flow language based on a Lustre semantics and is dedicated to real-time embedded software. It extends Lustre with features usually found in ML-languages such as typing and higher-order functions. It provides original features such as the arbitrary mix of data-flow and hierarchical automata [2] [68], various type-based static analysis [69], [70] and modular compilation into sequential code [60], [55] [54].

ReactiveML is an extension of Objective Caml with synchronous concurrency (based on synchronous parallel composition and broadcast of signals). The goal is to provide a general model of deterministic concurrency inside a general purpose functional language to program reactive systems (e.g., graphical interfaces, simulation systems). The research activity concerns the development of compilation techniques, dedicated type systems to ensure various safety properties (e.g., determinism, reactivity, boundedness) or the mix of both synchronous and asynchronous concurrency.

In collaboration with Albert Cohen and Christine Eisenbeis (INRIA Alchemy), Marc Duranton (Philips Natlabs, Eindhoven), we have introduced in 2005 a new programming model for the design of video intensive applications. This model, called the N-synchronous model, is based on an extension of the synchronous model allowing to combine non strictly synchronous streams provided that they can be synchronized through the use of bounded buffers. This is obtained by introducing particular clocks as infinite binary periodic words [111]. Thanks to the periodic nature of these clocks, we are able to verify properties like the absence of buffer overflows and deadlocks during the execution. Clock verification is expressed as a type-inference problem with a sub-typing rule. The core of the model has been settled in [65] and [66]. Florence Plateau works since that time on this subject. We introduced a notion of abstractions for these clocks in 2008 as a mean to reason about sets of (non necessarily periodic) clocks [67].

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. We have a collaboration with Gemalto in this area.

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).

5. Software

5.1. The CiME rewrite toolbox

Participants: Évelyne Contejean [contact], Claude Marché, 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*. This comes with a *Coq* library called Coccinelle (http://www.lri.fr/~contejea/Coccinelle). CiME3 and Coccinelle are also developed by the participants of the A3PAT project at the CNAM, and are distributed under the Cecill-C licence.

5.2. The Why platform

Participants: Jean-Christophe Filliâtre [contact], Romain Bardou, Claude Marché, Guillaume Melquiond, Yannick Moy, Christine Paulin-Mohring.

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. For C, we designed our own specification language, largely inspired from JML. Those are interpreted by the *Caduceus* tool.

The platform is distributed as open source, under GPL Licence, at http://why.lri.fr/.

A back-end tool also called *Why* serves as the VCG. It differs from other systems in that it outputs conditions for several existing provers: interactive ones (*Coq*, Isabelle/HOL, PVS, HOL-light, Mizar) and automatic ones (Simplify, *Alt-Ergo*, Gappa, and SMT provers Yices, CVC3, Z3, haRVey, etc.). The *Why* VCG alone has been used by external researchers in published verifications of non-trivial algorithms (Efficient square root used in GMP [53], Knuth's algorithm for prime numbers [108]).

Verification of Java Card applets using *Krakatoa* is under experimentation at Gemalto company. *Krakatoa* is also used for teaching (University of Evry, Ecole Polytechnique).

Caduceus is currently under experimentation at Gemalto company, at Dassault Aviation company, and at CEA (Saclay). It is also used for teaching at Ecole Polytechnique (2006/2007, 1st year master ISIC, *projet de verification*) and at University of Evry (2005-2006 and 2006-2007, proofs using *Coq*).

In 2007 and 2008, an Eclipse plugin for the platform has been developed (http://www.lri.fr/~oudot/).

5.3. The Alt-Ergo theorem prover

Participants: Sylvain Conchon [contact], Évelyne Contejean, Stéphane Lescuyer.

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

It is currently used in our team to prove correctness of C and Java programs as part of the *Why* platform. *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 distributed as open source, under the CeCILL-C licence, at URL http://alt-ergo.lri.fr.

5.4. Lucid Synchrone

Participant: Marc Pouzet [contact].

Lucid Synchrone is an experimental language for the implementation of reactive systems. It is based on the synchronous model of time as provided by Lustre combined with features from ML languages. It provides powerful extensions such as type and clock inference, type-based causality and initialization analysis and allows to arbitrarily mix data-flow systems and hierarchical automata or flows and valued signals.

It is distributed under binary form, at URL http://www.lri.fr/~pouzet/lucid-synchrone/.

The language has served as a laboratory to experiment various extensions of the language Lustre. Several programming constructs (e.g. merge, last) and type-based program analysis (e.g., typing, clock calculus) originaly introduced in Lucid Synchrone are integrated in the new SCADE 6 compiler developped at Esterel-Technologies.

5.5. Reactive ML

Participant: Louis Mandel [contact].

ReactiveML is a programming language dedicated to the implementation of interactive systems as found in graphical user interfaces, video games or simulation problems. ReactiveML is based on the synchronous reactive model due to Boussinot, embedded in an ML language (Objective Caml).

The Synchronous reactive model provides synchronous parallel composition and dynamic features like the dynamic creation of processes. In ReactiveML, the reactive model is integrated at the language level (not as a library) which leads to a safer and a more natural programming.

ReactiveML is distributed at URL http://www.lri.fr/~mandel/rml under the same licence as Ocaml (the compiler is distributed under the terms of the Q Public License and the library is distributed under the terms of the GNU Library General Public License). The development of ReactiveML started at the University Paris 6 (from 2002 to 2006).

The language is mainly used for the simulation of mobile ad hoc networks at the University Paris 6 and for the simulation of sensor networks at France Telecom and Verimag (CNRS, Grenoble).

5.6. Bibtex2html

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

Bibtex2html is a generator of HTML pages of bibliographic references. Distributed as open source since 1997, under the GPL licence, 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.

Last but not least, it is used by several INRIA teams for automatically producing their RAWEB bibliography.

5.7. 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 licence, at http://ocamlgraph.lri.fr/. It is also distributed as a package in several Linux distributions. Ocamlgraph is now widely spread among the community of Ocaml developers.

5.8. Mlpost

Participants: Jean-Christophe Filliâtre [contact], Johannes Kanig, 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 licence, at http://mlpost.lri.fr/. Mlpost was presented at JFLA'09 [18].

5.9. The Gappa tool

Participant: Guillaume Melquiond [contact].

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 [15].

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).

Gappa can also act as a backend for the *Coq* proof assistant in order to prove properties related to the certification of floating-point programs [20]. In 2009, the support of Gappa as an automated prover for the *Why* system has also been greatly improved.

5.10. The Interval package for Coq

Participant: Guillaume Melquiond [contact].

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/. Most of the work on this library was done while in the Mathematical Components team (Microsoft Research–INRIA Joint Research Center).

6. New Results

6.1. Randomized programs

Participant: Christine Paulin-Mohring.

The work of C. Paulin and Ph. Audebaud from ÉNS Lyon for modeling probabilistic programs in *Coq* as probability measures has been published in a special issue of the journal Science of Computer Programming [12]. A new version of the corresponding *Coq* library [102] has been written which uses the new mechanism of type classes in *Coq* designed by M. Sozeau and N. Oury and includes a few automated tactics. It is based on a general theory of ordered sets and cpos [30] and contains high-level theorems for analysing recursive programs. It is currently used in Verimag (Grenoble) and in the Marelle INRIA team (Sophia-Antipolis) as the basis for the CertiCrypt environment for formalizing proofs in computational cryptography in the framework of the SCALP project. We participate to the formalisation in *Coq* of the Computational Indistinguishability Logic (CIL) designed in the project. This work has been presented at the conference dedicated to Per Martin-Löf on the occasion of his retirement in Uppsala in May.

6.2. Floating-point algorithms and proofs

Participants: Sylvie Boldo, Guillaume Melquiond.

An established collaboration with M. Daumas (Université of Perpignan Via Domitia) and R.-C. Li (University of Texas at Arlington, USA) with S. Boldo has been carried on about argument reductions [14] and has been published in IEEE Transactions on Computers. This is the first step and a very delicate one to compute elementary functions (exponential, sine...).

S. Boldo and G. Melquiond developed with P. Zimmermann (CACAO, INRIA Lorraine) and S. Rump (Institute for Reliable Computing, Hamburg) a simple and efficient method to compute and/or estimate the predecessor and successor of a floating-point number using only floating-point operations in rounding to nearest [16].

S. Boldo, J.-C. Filliâtre and G. Melquiond implemented a mechanism for calling Gappa (an automatic tool specialized in floating-point arithmetic) from a *Coq* interactive proof. This offers a significant speedup in the process of verifying floating-point programs [20].

G. Melquiond participed to the writing of a handbook on floating-point arithmetic [29] directed by Jean-Michel Muller (Arenaire, INRIA Rhône-Alpes).

6.3. Tools for the working scientist

Participants: Romain Bardou, François Bobot, Sylvain Conchon, Jean-Christophe Filliâtre, Johannes Kanig.

The Mlpost tool developped by R. Bardou, F. Bobot, J.-C. Filliâtre, J. Kanig and S. Lescuyer was presented at JFLA'09 [18].

S. Conchon and J.-C. Filliâtre supervised Ocamlviz, a summer project funded by Jane Street Capital (NYC, USA) and worked out by two graduate students from Université Paris-Sud. Ocamlviz is a set of tool to perform real-time monitoring of Ocaml programs. Ocamlviz is to be presented at JFLA 2010 [75].

6.4. Proof of imperative and object-oriented programs

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Paolo Herms, Johannes Kanig, Kalyan Krishnamani, Claude Marché, Yannick Moy, Thi-Minh-Tuyen Nguyen, Christine Paulin-Mohring, Asma Tafat Bouzid, Wendi Urribarrí.

6.4.1. The Why and Frama-C tools

J.-C. Filliâtre gave an invited talk at AFM'09 (a CAV'09 workshop in Grenoble, France) [17]. This talk was a tutorial on the *Why* tool, showing how to use it to prove algorithms and as an intermediate language in the process of verifying programs written in languages such as C. J.-C. Filliâtre was later invited at the National Institute of Aerospace (Hampton, USA) and at AdaCore (France) to give the same tutorial.

The cooperation with CEA List around the design of the Frama-C leads us to the description of a proposal for a general-purpose specification language called ACSL (ANSI/ISO C Specification Language). The fourth version, numbered 1.4, is published as a technical report [32] with significant contributions from C. Marché, Y. Moy and J.-C. Filliâtre.

The *Jessie* intermediate language serves as a new target language of translation from C and Java, and is itself translated in a second stage into the input language of the *Why* VCG.

Y. Moy developed a plugin to connect *Jessie* to the *Frama-C* platform, and this plugin is distributed since the "Lithium" release of Frama-C in October 2008. Since the "Beryllium-2" release of Frama-C in October 2009, *Jessie* is distributed independently of Frama-C, as a dynamic plugin [33]. The *Jessie* plugin is now the main tool for adding new experimental features coming from new theoretical studies as described below.

A. Paskevich augmented the input logical language of *Why* with algebraic polymorphic types and pattern matching expressions [34]. For automated SMT provers that do not support algebraic types and match expressions, translation into first-order language is implemented. An immediate application of this work is to facilitate handling of enumerated types which are particularly useful in verification of floating-point computations.

6.4.2. Floating-Point C Programs

S. Boldo and J.-C. Filliâtre introduced in 2007 [1] annotations specific to floating-point arithmetic, that have been successfully applied to several floating-point programs. C. Marché and A. Ayad continued this work [31], [37] by developing a floating-point extension of the Jessie plugin of the Frama-C platform in order to handle numerical C programs. With respect to the former approach of Boldo-Filliâtre, first it supports full floating-point arithmetic, that is involving special values for infinity and Nan (not-a-number); and second it allows to call automatic provers for discharging proof obligations. It has been applied to a formal verification of a C library for interval arithmetic [37].

Professor William Kahan proposed in November 2004 a program for a precise discriminant for quadratic equations. Proofs were described as "far longer and trickier" than the algorithms and programs and the author deferred their publication. S. Boldo has done a full formal proof of the program, including the fact that the main test in the program can be wrong due to floating-point errors. This is published in IEEE Transactions on Computers [13].

As part of the FOST project, S. Boldo and J.-C. Filliâtre study a real-life program computing the discretization of the spread of acoustic waves on a rope. S. Boldo has moreover developed an appropriate technique to bound the rounding error when computing a sequence of values. In our case, a naive approach would give an error proportional to the exponential of the number of steps. The idea is to give an analytical expression of the rounding error. This technique was successfully applied to a second order linear recurrence where the error was proved proportional to the square of the number of steps and the same result is expected for a function part of the computation of the gradient. An article describing this application and this technique has been published in ICALP [19].

S. Boldo and T. Nguyen designed an approach that states the rounding error whatever the execution hardware and the compilation. The goal is to formally prove, in the Frama-C platform, properties about numerical programs that are true for multiple architectures and compilers. A submitted article describes this technique and an avionics application.

6.4.3. Automatic generation of annotations

When applying deductive verification techniques to industrial programs, a key issue identified was the need of an automatic synthesis of annotations.

A general and systematic approach has been studied by Moy, based on the known successful techniques based on abstract interpretation. Classically, forward abstract interpretation can be used to discover programs invariants, in particular on integer variables, by analyzing a program as a whole. We proposed new techniques in order to analyze program procedures independently, by a contextual analysis. Moy proposed to combine abstract interpretation, weakest precondition calculus and quantifier elimination. This is part of Moy's PhD thesis [11]; submitted to a journal.

Y. Moy worked on providing guarantees about the memory safety of real C programs used in embedded devices. This originated in a need expressed at France Télécom R&D that no available tool could fulfill. He focused on designing modular, contextual and idiomatic approaches to memory safety for C pointer programs. He showed how memory safety can be reduced to checking assertions and how the automatic generation of annotations presented above can be used to propagate backward those assertions in function preconditions. Experiments have been performed on several libraries of C code, and results are successful in two ways: first, a significant number of bugs have been discovered (reported back to the authors, with patches, all patches have been accepted), and patched code has almost been proved correct (99% of the VCs). This is reported in Y. Moy's PhD thesis [11].

Y. Moy also proposed a new approach for supporting the union types and the pointer casts of C programs, in the *Why* platform. It consists of a global analysis of the C source code to detect all possible pointer casts and generate accordingly an intermediate *Jessie* program. This is implemented in the *Jessie* plugin of the *Frama-C* platform, and is described in a chapter of Y. Moy PhD thesis.

In his post-doc started in Sep. 09, K. Krishnamani investigates the possible use of predicate abstraction techniques for automatically generating annotations. He continued to study the combination of BDDs and SMT for predicate abstraction: traditionally, quantifier elimination techniques within automated abstraction-refinement frameworks (*CEGAR*) work by employing SAT solvers for performing quantifier elimination. It is a well known fact that BDDs are very efficient for performing the same, when we are in the domain of Boolean-encoded systems. When we move to the more challenging domains of Hybrid systems (which have Boolean control - discrete transitions, as well as continuous transitions) quantifier elimination using pure BDD algorithms becomes infeasible. In order to accomplish this, BDD algorithms are integrated in SMT reasoning that handles various theories of interest (reals in the case of hybrid systems). In a joint work with A. Cimatti, A. Franzen, A. Griggio and M. Roveri [63], K. Krishnamani improves on [61] by making use of various features of SMT solvers (backjumping, early pruning, and several other optimizations); and on the BDD side, an array of BDDs (conjunctively partitioned BDD) instead of a monolithic BDD, and look-ahead mechanisms to identify inconsistencies earlier during the search.

6.4.4. Modularity, Abstraction, Separation and Refinement

When dealing with large programs, it is essential to provide mechanisms for modularity and abstraction of data-types. We have started several directions of research around the *Why* platform, to provide appropriate abstraction mechanisms at different levels.

Wendi Urribarrí in her thesis supervised by Christine Paulin-Mohring proposed a module system for the input language of the *Why* VCG. She defined notions of modules and interfaces, including functors and she proposed refinement rules between interfaces. She proved a general principle of state variables hiding. She is currently implementing these results in the *Why* platform. This work is presented in a submitted paper [44].

R. Bardou's PhD thesis is aimed at the support of structure invariants in reasoning on pointer programs. He designed a new type system for this purpose, inspired on one hand by memory-management techniques based on *regions* [107], [109] and on *permissions* [83], and on the other hand techniques to guarantee invariant preservation based on ownership [64], [49]. The resulting language and type system allows to guarantee invariant preservation by static typing, whereas former approaches required theorem proving [49] or dynamic checking [48]. This is reported in an article under submission [38].

F. Bobot started a PhD thesis on September 2008 on the combination of traditional separation logic and Burstall-Bornat memory models such as the ones used in the *Why* platform for the verification of C and Java programs. The idea is to combine the benefits of both techniques: separation for free when it is statically known from the code that two pointers cannot be aliased, possibly using static analysis as described above; and ability for the user to express pointer separation in other cases with the connectives of separation logic, possibly combined with inductive predicates to tackle tree-like data structures. An implementation has been done in the *Jessie* tool. It allows to generate automatically separation predicates from inductive predicates. Afterward, the user can use them in his specifications. This work have been presented at the COST meeting at Eindhoven.

With C. Hurlin (INRIA Sophia Antipolis) and A. J. Summers (Imperial College London), F. Bobot proposed techniques to disprove entailment of intuitionistic and classical separation logic. These techniques have been certified in Coq [24].

With E. Tushkanova, A. Giorgetti and O. Kouchnarenko at LIFC Besançon, C. Marché proposed extensions to specification languages in order to specify modularly *generic* Java programs [35]. Such programs make use of type parametricity and higher-order constructions, and specifying such programs need to add new notions in specification languages: notion of parametric theory, notion of theory instantiation.

In her Master's internship [43], A. Tafat proposed a new refinement technique for object-oriented programs. The main idea was to combine the Spec# ownership system [49] with refinement techniques of the B method [58]. Furthermore, the approach allows to hide side-effects on private data of classes. An article currently submitted [42] has been written with C. Marché and S. Boulmé (VERIMAG, Grenoble).

6.4.5. Higher-order programs

We are also interested in tackling ML programs, in particular within the ARC CeProMi. One of the key difficulties of proving ML programs is the mix of higher-order features and side effects. J. Kanig and J.-C. Filliâtre proposed an extension of the *Why* system in which one can specify higher order programs and obtain proof obligations, expressed in Higher-Order Logics. They also presented an implementation of this development [25]. A number of case studies have been realized using this tool, in particular a proof of an implementation of the Koda-Ruskey algorithm [90], a program which heavily relies on both side effects and higher-order features. This is the first time this particular algorithm has been proved correct.

6.5. Automated Deduction

Participants: Sylvain Conchon, Évelyne Contejean, Jean-Christophe Filliâtre, Mohamed Iguernelala, Stéphane Lescuyer, Claude Marché, Alain Mebsout, Guillaume Melquiond, Xavier Urbain.

6.5.1. Formalization of an efficient SAT-solver

In an attempt to improve automation capabilities in the Coq proof assistant, Conchon and Lescuyer developed a tactic for the propositional fragment based on the DPLL procedure. Although formulas naturally arising in interactive proofs do not require a state-of-the-art SAT solver, the conversion to clausal form required by DPLL can strongly damage the performance of the procedure. In [26], we have presented a reflexive DPLL algorithm formalized in Coq which outperforms the existing tactics. It is tightly coupled with a lazy CNF conversion scheme which, unlike Tseitin-style approaches, does not disrupt the procedure. This conversion relies on a lazy mechanism which requires slight adaptations of the original DPLL. As far as we know, this is the first formal proof of this mechanism and its Coq implementation raised interesting challenges.

6.5.2. Matching

Arthur Milchior did an undergraduate internship under the supervision of Sylvain Conchon and Jean-Christophe Filliâtre from June to August 2009. He improved a matching algorithm from L. de Moura and N. Bjørner and implemented it in the *Alt-Ergo* theorem prover [41]. More precisely, the improvement consisted in handling types in the matching compilation to take polymorphism into account. The experimental results showed a 30% speed-up with respect to the current matching implementation.

6.5.3. Integration of associative-commutative symbols in Alt-Ergo

Associative and commutative (AC) symbols are ubiquitous in mathematics and in the modelling of data structures (*e.g.* multisets). Handling AC via axiomatization, as done in *Alt-Ergo* before the internship of Iguernelala, is hightly inefficient and incomplete. Iguernelala, together with Conchon and Contejean, has proposed an extension of the congruence closure modulo an arbitrary theory at the core of *Alt-Ergo* to handle also (arbitrarily used-defined) AC symbols [39]. This extension is integrated to the distributed version of *Alt-Ergo* and enables, for instance, the combination of AC, linear arithmetics, and the theory of equality (congruence closure).

6.5.4. The Alt-Ergo theorem prover

Based on our experience with the development of *Alt-Ergo*, we have shown a small number of modifications needed to bring parametric polymorphism to our SMT solver [56]. The first one occurs in the typing module where unification is now necessary for solving polymorphic constraints over types. The second one consists in extending triggers' definition in order to deal with both term and type variables. Last, the matching module must be modified to account for the instantiation of type variables.

Concerning the prover itself, we fully formalized the core decision procedure $CC(X)$ of *Alt-Ergo* in the *Coq* proof assistant. Moreover we provided a formal proof of soundness and completeness [71].

6.5.5. Data structures

S. Conchon and J.-C. Filliâtre generalized an idea present in the work described in [73], [72] and introduced the new notion of *semi-persistence*. A data structure is said to be semi-persistent when only the most recent version and its ancestors can be accessed or updated. Making a data structure semi-persistent may improve its time and space complexity. This is of particular interest in backtracking algorithms manipulating persistent data structures, where this property is usually satisfied. In particular, this is the case for the union-find data structure used internally by *Alt-Ergo*. S. Conchon and J.-C. Filliâtre proposed a proof system to statically check the valid use of semi-persistent data structures. It requires a few annotations from the user and then generates VCs that are automatically discharged by a dedicated decision procedure. An article was presented at ESOP'08 [74].

6.5.6. Automated proofs and certificates

6.5.6.1. Coccinelle and CiME's traces

The powerful subterm criterion for termination of term rewriting systems is now part of the certified criteria in CiME\Coccinelle. In addition to providing its first *Coq* formalisation, É. Contejean, A. Paskevich, and X. Urbain extended the subterm criterion by weakening its premises [78]. This is a joint work with J. Forest (ENSIIE), P. Courtieu, and O. Pons (CNAM).

To facilitate interactions between other provers and certifying engines, the CiME\Coccinelle team was involved in the definition of a certification problem format (CPF, http://cl-informatik.uibk.ac.at/software/cpf/) the major design of which was done in collaboration with the CeTA\IsaFoR group in Innsbruck, and with additional comments from the CoLoR group in Beijin.

6.5.6.2. Proofs of bounds on real-valued expressions

G. Melquiond has built a library for automatically proving bounds on expressions in the *Coq* system [99], [100]. This library performs automatic differentiation and interval arithmetic (with floating-point bounds). Its purpose is to help the user with the mathematical part of the certification of numerical programs.

6.6. Synchronous Programming

Participants: Louis Mandel, Florence Plateau, Marc Pouzet.

6.6.1. Modular Static Scheduling of Synchronous block-diagram

In collaboration with P. Raymond (VERIMAG, Grenoble), we have worked on the modular generation of sequential imperative code from synchronous data-flow networks. Precisely, given a system with several input and output flows, how to decompose it into a minimal number of classes executed atomically and statically scheduled without restricting possible feedback loops between input and output? Though this question has been identified in the early years of Lustre, it has almost been left aside until the recent work of Lublinerman, Szegedy and Tripakis presented at POPL'09. The problem is proven to be intractable and the authors derive an iterative algorithm looking for solutions for c = 1,2,... classes where each step is encoded as a SAT problem. Despite the apparent intractability of the problem, it appears that real problems rarely fall in this category. Based on an analysis of input/output dependences, we have proposed a polynomial algorithm which (1) either gives an optimal solution or (2) gives a non trivial lower bound on the number of classes to start an iterative combinatorial search. In all the examples we have considered (the whole SCADE library and two industrial examples), the polynomial algorithm finds an optimal scheduling.

This works has been presented at EMSOFT'09 [28] and was nominated among the three best papers.

6.6.2. Objects in Synchronous Block-diagrams

In collaboration with Paul Caspi, Pascal Raymond (VERIMAG, Grenoble), Jean-Louis Colaço (Prover Technologies), we have proposed an extension of a synchronous language with a classed-based mechanism. The problem we addressed is the modular specification and implementation of systems with *modes*. Typical applications involve different control laws corresponding to different phases, e.g., take-off, full flight and landing in a fly-by-wire control system. On one hand, existing methods such as the combination of Simulink/Stateflow provide powerful but unsafe mechanisms by means of imperative updates of shared variables. On the other hand, synchronous languages and tools such as Esterel or Scade/Lustre are too restrictive and forbid to fully separate the specification of modes from their actual instantiation with a particular control automaton.

We introduce a conservative extension of a synchronous data-flow language close to Lustre, in order to be able to define systems with modes in a more modular way, while insuring the absence of data-races. We show that such a system can be viewed as an object where modes are methods acting on a shared memory. The object is associated to a scheduling policy which specifies the ways methods can be called to build a valid synchronous reaction. We show that the verification of the proper use of an object reduces to a type inference problem using row types introduced by Wand, Rémy and Vouillon. We define the semantics of the extended synchronous language and the type system. This work has been presented at LCTES'09 [21].

During the year, a new compiler — mainly the V4 of Lucid Synchrone — has been rebuilt on this paradigm.

6.6.3. Alternative synchronous models

The N-synchronous model introduced a way to compose streams which have *almost the same clock* and can be synchronized through the use of a finite buffer. This relaxed model of synchrony is achieved by introducing a subtyping rule in the clock type system to localise places where some synchronization code must be inserted. Subtyping can be checked when clocks are defined by ultimately periodic infinite binary words. Nevertheless, this check can be costly if the patterns are long, and is in any case restricted to periodic behaviors only.

To achieve this relaxed model on non periodic clocks, the notion of *clock envelopes* has been introduced. These envelopes are abstractions of clocks that gives bounds on the number of instants where the stream has been present since the beginning of the execution. An envelope is a set of (not necessarily periodic) clocks and subtyping can be checked by simple arithmetic comparison. A first abstraction has been presented in APLAS [67]. It has been improved and presented at JFLA 2009 [27] and HFL 2009 [22].

Most of the algebraic properties of this new abstraction have been formalized and proved in the proof assistant *Coq* (http://www.lri.fr/~plateau/hfl09/).

An implementation called Lucy-n has been developped during the year. The language is a Lustre-like language extended with a buffer construct. It is based on a relaxed clock-calculus where the compiler automatically infers the size of buffers. Various techniques to solve constraints have been implemented (e.g., exact resolution method and abstract method based on the technique presented in [22]).

F. Plateau will defend her PhD. Thesis on the N-synchronous model in January 2010.

L. Mandel, F. Plateau and M. Pouzet have worked on the relaxed model of synchrony during the period [67].

6.6.4. Compiler certification

The work on the development of a certified compiler for a synchronous language has been pursued in 2008. We have formalized in Coq the semantics of the source and target language (work in collaborartion with Grégoire Hamon and Jean-Louis Colaço).

A prelimary version of this work presenting a minimal formalization of a compiler is presented in [54].

6.6.5. Dynamic aspects in synchronous languages

We have continued the development of ReactiveML, an extension of Objective Caml with reactive constructs [95]. A tool demo has been done during the ML workshop 2009.

In addition, a toplevel for Lucid Synchrone has been proposed. It provides a way to interactively define synchronous data-flow programs and thus dynamically add new processes to execute. A new communication scheme through shared channels has also been proposed.

7. Contracts and Grants with Industry

7.1. System@tic: PFC

Participants: Jean-Christophe Filliâtre, Claude Marché, Christine Paulin-Mohring, Nicolas Stouls.

The PFC project (Plateforme de Confiance, trusted platforms) is a project in the SYSTEM@TIC Paris Region French cluster in complex systems design and management http://www.systematic-paris-region.org. This cluster involves industrial groups, SMEs and academic partners in the Paris-Region and is supported by the french government and the regional council.

The goal of the project is the conception and validation of secure and safe embedded applications. Within this project, we closely collaborate with Gemalto, CEA-LIST and Trusted Logic.

This project is funded by the french ministry of industry (FCE) for 30 month from December 2006 to May 2009.

7.2. CEA-Airbus contract

Participants: Sylvain Conchon, 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. CIFRE grants

Participants: Claude Marché, Yannick Moy, Christine Paulin-Mohring.

CIFRE, France Télécom R & D Yannick Moy started his PhD in January 2006. He is working on the analysis of the usage of dynamic memory in embedded C code. The goal is to design static analysis methods for pointer manipulations specialized to the telecommunications applications. The thesis has been defended in January 2009.

8. Other Grants and Activities

8.1. National initiatives

8.1.1. CAT

Participants: Jean-Christophe Filliâtre, Claude Marché.

CAT (C Analysis Tools) is a RNTL project related to the verification of C programs. It started in June 2006 and ended in August 2009.

The goal of the project was to develop an open-source toolkit for analysing industrial-size C programs during development, verification, maintenance and evolution. The resulting environment is called Frama-C (http://frama-c.cea.fr)

Our partners: CEA List (Saclay, project leader), INRIA Rennes (Team Lande), Dassault Aviation (Saint-Cloud), Airbus France (Toulouse), Siemens.

8.1.2. U3CAT

Participants: Jean-Christophe Filliâtre, Claude Marché, Marc Pouzet, 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).

8.1.3. A3PAT

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer, Andrei Paskevich.

A3PAT (Assister Automatiquement les Assistants de Preuve Avec des Traces, Helping proof assistants with full automation by means of traces, literally "on three legs") is a project funded by ANR, started in December 2005 and ended in May 2009. http://a3pat.ensiie.fr/.

It aimed at helping proof assistants with trustworthy decision procedures, in particular by generating proof traces in order to build proof terms.

The principal investigator was Xavier Urbain (ENSIIE). The scientific leaders were Yves Bertot (Inria Sophia), Pierre Casteran (Labri, Bordeaux 1), and Évelyne Contejean (LRI, Orsay).

Contributions of the project were:

- A solution for the certification of automated proofs in the context of first order term rewriting systems. This solution is based on a huge *Coq* library for rewriting: Coccinelle, and a certification engine, part of the CiME 3 rewriting tool [79], [78]. It involves some of the most advanced techniques for termination, amongst which some unique features; its development provided new proof techniques [81] as well as new termination criteria [82], [78]. Automated provers can certify their proofs with solution, using an XML language well-suited for proof traces.
- Several formal libraries, in particular on ordinals representation in *Coq*.

Deliverables included publications, software, and libraries.

8.1.4. ADT Alt-Ergo

Participants: Sylvain Conchon, 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).

8.1.5. ARC CeProMi

Participants: Claude Marché, Jean-Christophe Filliâtre, Christine Paulin-Mohring, Wendi Urribarrí, Johannes Kanig, Romain Bardou, Asma Tafat.

CeProMi (Certification de Programmes manipulant la Mémoire) is a *ARC* (Action de Recherche Collaborative), a 2-years project funded by INRIA, started in January 2008, and ended in December 2009 http://www.lri.fr/cepromi/

The goal was to propose new theoretical bases for proving programs involving memory sharing and side effects (typically, pointer programs in C, objects in OO languages, records with mutable fields in ML).

There were three different levels of studies: extensions of specification languages with appropriate notions of invariants and description of side effects; design of advanced type systems and static analyses for detecting either alias or separation of pointers; design of verification conditions calculi incorporating notions of modules, pointer separation and refinement.

Partners: Gallium team (INRIA Rocquencourt), Cassis team (INRIA Nancy), TFC team (LIFC, Besançon), DCS team (VERIMAG, Grenoble)

8.1.6. CerPAN

Participants: Sylvie Boldo, Jean-Christophe Filliâtre, Ali Ayad.

CerPAN (Certification de Programmes d'Analyse Numérique) is a 3 years project funded by ANR, started in December 2005 that has been extended until June 2009. http://www-lipn.univ-paris13.fr/CerPAN/

This project aimed at developing and applying methods which allow to formally prove the soundness of programs coming from numerical analysis techniques. We were more precisely working on problems related to the verification of floating point algorithms. The partners were: University Paris 13, INRIA and CNAM.

8.1.7. FOST

Participants: Sylvie Boldo, 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).

8.1.8. Hisseo

Participants: Sylvie Boldo, Claude Marché, Guillaume Melquiond, Thi-Minh-Tuyen Nguyen, Ali Ayad.

Hisseo is a 3 years Digiteo project that started in September 2008. http://hisseo.saclay.inria.fr

The Hisseo project will focus 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).

8.1.9. Pactole

Participants: Évelyne Contejean, Jean-Christophe Filliâtre, Xavier Urbain.

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).

8.1.10. SIESTA

Participant: Marc Pouzet.

SIESTA is a 4 year project funded by ANR RNTL. The coordinator is Y. Parissis (LIG, Grenoble). http://www.siesta-project.com. The project started in january 2008.

This project addresses the automated testing of embedded systems implemented in SCADE or Simulink. M Pouzet is involved on the architecture of the SCADE 6 compiler to integrate verification techniques. The challenge is to take new programming constructs (e.g., hierarchical automata, reset and general forms of clocks) into account to improve verification techniques and modularity.

Partners: AIRBUS, Turbomea, Hispano-Suiza, Onera, CEA List, Esterel-Technologies, EADS, LRI, LIG.

8.1.11. GENCOD

Participant: Marc Pouzet.

This project is funded by DGE for two years. The coordinator is Thales. The project started in sept. 2008 (delayed by one year).

The GENCOD project aims at defining methods to certify the Esterel compiler for hardware (Norm. DO 254, the hardware version of DO 178 B used for critical software).

Partners: Dassault-Aviation, Thales, Esterel-Technologies, TNI, Airbus, LRI.

8.1.12. INRIA Action d'Envergure Synchronics

Participants: Marc Pouzet, Louis Mandel.

This project is funded by INRIA for 4 years and started in Jan. 2008. The coordinators are A. Girault (INRIA Rhône Alpes) and M. Pouzet. http://synchronics.wiki.irisa.fr/

The goal of the project is to propose new languages for the development of embedded systems allowing *from a unique source* to both simulate the system with its environment and generate code. It capitalizes on recent extensions of data-flow synchronous languages (Lucid Synchrone, ReactiveML), a relaxed form of synchrony, and means to mix discrete and continous systems inside the synchronous model of time.

The project focuses on language extensions to increase modularity, dedicated type systems to ensure safety properties, efficient compilation and the mix of discrete and continuous time.

Partners: INRIA Rhône Alpes (Alain Girault, Bertrand Jeannet), IRISA (Benoit Caillaud), VERIMAG (Erwan Jahier, Pascal Raymond), INRIA Saclay (Albert Cohen, Marc Pouzet, Louis Mandel)

8.1.13. SCALP

Participants: Christine Paulin-Mohring, Yann Régis-Gianas, 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.

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

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

8.1.14. PARTOUT

Participants: Louis Mandel, Marc Pouzet.

This project is funded by ANR (program DEFIS).

It started on january 2009 for 4 years; the coordinator is Frédéric Boussinot from INRIA Mimosa.

Partners: INRIA Mimosa, CNAM, LRI.

The goal of the project PARTOUT is, from a programming language point of view, to study the impact on programming of the globalization of parallelism which now covers all the spectrum of informatics, ranging from multicore architectures and distributed systems, up to applications deployed on the Web.

8.1.15. 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

8.2. European initiatives

8.2.1. European COST action FVOOS

FVOOS (Formal Verification of Object-Oriented Programs, http://www.cost-ic0701.org/) is a COST (European Cooperation in the field of Scientific and Technical Research, http://www.cost.esf.org/) action. It started in 2008 and will last until April 2011.

It involves 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.

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.

8.3. International initiatives

8.3.1. Taiwan

We have an ORCHID project with the National Taiwan University (Taipei, Taiwan) on Formal Methods for Software Security.

8.4. Visits, researcher invitation

8.4.1. Visits

J.-C. Filliâtre visited Alwyn Goodloe at the National Institute of Aerospace (Hampton, USA) in September 2009 and gave a tutorial on the *Why* tool. J.-C. Filliâtre also presented the ACSL specification language [32] and the Frama-C platform.

M. Pouzet visited Arvind (MIT, Boston, USA) and Grégoire Hamon (TheMathworks, USA) in July 2009. He presented the N-synchronous model.

8.4.2. Invitations

Barbará Vieira, Ph.D. student at Universidade do Minho (Braga, Portugal), visited ProVal from March to May 2009. She worked with J.-C. Filliâtre on a verification tool for CAO, a domain-specific language for cryptographic protocols (http://www.cs.bris.ac.uk/~page/research/cao.html).

9. Dissemination

9.1. Interaction with the scientific community

9.1.1. Prices and distinctions

Since 2007, Marc Pouzet is a junior member of the IUF (*"Institut Universitaire de France"*), that distinguishes each year a few French university professors for the high quality of their research activities.

The Ocamlviz summer project [75] was a great success, according to Jane Street's managing director Yaron Minsky:

> << From my point of view, the single most useful project is unquestionably ocamlviz. Ocamlviz is a realtime profiling tool for OCaml, and I was really impressed with the system's polish. The design is carefully thought out; it seems to be quite well implemented; the front-end has a surprisingly usable UI; there's a nice looking website for it, and good documentation to boot. It's really a fantastic effort, and I expect we'll be taking it for a spin on some of our own OCaml projects. >>
>
> Yaron Minsky, Jane Street Capital
>
> http://janestcapital.com/?q=node/68

Project Pactole won the best Digiteo poster award at the annual Digiteo forum on the 21st of October http://www.digiteo.fr/Digiteo_Annual_Forum_2009.

9.1.2. Collective responsibilities within INRIA

S. Boldo was elected representative of the researchers at the "comité de centre" of the INRIA Saclay - Île-de-France (2008–2010).

C. Paulin is *déléguée scientifique* of the centre INRIA Saclay - Île-de-France and she is a member of the national evaluation board of INRIA. In 2009, she participated to the hiring committee of CR2 positions at INRIA Sophia-Antipolis.

9.1.3. Collective responsibilities outside INRIA

C. Marché is a member of the *"commission consultative de spécialistes"*, Section 27, University Paris-Sud 11, since Sep. 08. In 2009, he participated to the hiring committee of one assistant professor position at IUT Orsay, and to the hiring committee of two assistant professor positions at UFR Orsay.

C. Marché is the French National Coordinator for the COST action "Formal Verification of Object-Oriented Programs" (2008-2011).

C. Marché is member 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/, since April 2007.

M. Pouzet was a member of the hiring committee for a Professor in computer science, section 27, INPL Nancy, spring 2009. He was member of a hiring committee for an Assistant Professor at Universié Nancy I and for an Assistant Professor at ENSIMAG, Grenoble.

C. Paulin participated to the hiring committees of two assistant professor positions at ENS Cachan and at the Ecole des Mines de Nantes, and to the hiring committee of one professor position at ENS Lyon.

C. Paulin is director of the Graduate school in Computer Science at University Paris Sud http://dep-info.u-psud.fr/ed/.

É. Contejean is a member of the *"jury de l'agrégation externe de mathématiques"* as an expert in computer science since 2007.

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.

G. Melquiond is an elected officer of the IEEE-1788 standardization committee on interval arithmetic since 2008.

S. Conchon is a member of the *"commission consultative de spécialistes"*, Section 27, University Paris-Sud 11, since Sep. 08. In 2009, he participated to the hiring committee of one assistant professor position at IUT Orsay.

S. Conchon is *correcteur au concours d'entrée á l'École Polytechnique* (computer science examiner for the entrance exam at École Polytechnique) since 2009.

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).

S. Conchon is an elected member of the board (*"conseil du laboratoire"*) of Laboratoire de Recherche en Informatique (LRI) since 2004.

M. Pouzet is an elected member of the *"conseil du laboratoire"* of LRI since 2006.

9.1.4. Event organization

J.-C. Filliâtre co-organized (with Leo Freitas, University of York, UK) the VSTTE'09 workshop (November 2009, Eindhoven, the Netherlands). http://vstte09.lri.fr/.

J.-C. Filliâtre is co-organizing (with Cormac Flanagan, University of California, Santa Cruz, CA, USA) the PLPV'10 workshop (January 2010, Madrid, Spain). http://slang.soe.ucsc.edu/plpv10/.

C. Paulin co-organized the *Digiteo Annual Forum 2009* (http://www.digiteo.fr/Digiteo_Annual_Forum_2009) which happened on Oct 21, 2009 at École Polytechnique. 335 participants. C. Marché was responsible for a session on critical software at this Forum.

9.1.5. Editorial boards

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

Marc Pouzet is associate editor of the EURASIP Journal on Embedded systems (http://www.hindawi.com/journals/es/. He is "directeur de collection" for Hermes publisher.

9.1.6. Program committees

C. Marché was a member of the program committee of the 22th International Conference on Automated Deduction (CADE'09).

C. Paulin is a member of the program committee of the 10th International Conference on Mathematics of Program Construction (MPC 2010).

S. Conchon is the vice-president of the "Journées Francophones des Langages Applicatifs" (JFLA) 2010.

J.-C. Filliâtre was a member of the program committees of AFM 2009, PLMMS 2009, TPHOLs 2009, VSTTE 2009 and PLPV 2010.

M. Pouzet was member of the program committee of the Real-Time and Network Systems Conference 2009 and MSR 2009. He was member of the program committe of the HFL (Hardware Functional Languages) 2009 workshop.

9.1.7. Invited Presentations

J.-C. Filliâtre was invited speaker at AFM'09 (Grenoble, France): *Invited tutorial: Why — an intermediate language for deductive program verification* [17].

M. Pouzet was invited speaker at "journées de l'AFSEC, Toulouse, 27 janvier 2009: *Abstraction d'horloges dans les systèmes synchrones*.

M. Pouzet was invited to the annual meeting of IFIP WG2.8 on Functional Programming (Frauenchiemsee, Germany, June 7-12, 2009).

S. Boldo was invited to the main INRIA Paris-Rocquencourt seminar, *"Le modèle et l'algorithme"* (http://www.inria.fr/rocquencourt/rendez-vous/modele-et-algo/), on October 1st.

C. Paulin was invited to the conference *"Philosophy and Foundations of Mathematics : Epistemological and Ontological Aspects"* dedicated to Per Martin-Löf on the occasion of his retirement in Uppsala, May 5-8, 2009.

G. Melquiond was invited to present the IEEE-1788 standardization process [23] at the Arith 19 symposium in Portland, OR, June 8-10, 2009.

G. Melquiond was invited speaker at the arithmetic workshop of GDR IM (RAIM'09) in Lyon, October 26-28, 2009.

9.1.8. PhD theses defended

Yannick Moy defended his PhD on January 15th, 2009. He was supervised by C. Marché (CIFRE thesis co-supervised by P. Crégut, France-Telecom). He is now senior software engineer at the AdaCore Company, at the European Office in Paris. (http://www.adacore.com).

9.1.9. Participation to PhD juries

C. Marché was president of the PhD jury of Clément Hurlin (University of Nice, September 14th, 2009).

C. Paulin was a member of the PhD jury of Zainah Dargaye (Univ. Paris 7, July 2009) and Benoît Razet (Univ. Paris 7, November 2009).

M. Pouzet was member of the PhD jury of Carlos Olarte (Ecole Polytechnique, september 2009).

Jean-Christophe Filliâtre reviewed the manuscript and was a member of the PhD jury of Benoît Razet (Univ. Paris 7, November 2009).

9.2. Teaching

9.2.1. Supervision of PhDs

S. Boldo and C. Marché are supervising the Ph.D. thesis of T. Nguyen that started in February 2009 as part of the Hisseo project (static analysis of the assembly code).

S. Conchon and É. Contejean are supervising the Ph.D. thesis of Stéphane Lescuyer (started September 2007) (complete certification of an automated theorem prover dedicated to program verification) and the Ph.D thesis of Mohamed Iguernelala (started September 2009) (forward and backward strategies in SMT solvers).

J.-C. Filliâtre is supervising the Ph.D. thesis of Johannes Kanig (started September 2007) and the Ph.D. thesis of François Bobot (started September 2008).

C. Marché is supervising the PhD theses of Romain Bardou since Sep. 07 (modular reasoning on pointer programs); of Asma Tafat since Sep. 09 (dynamic invariants); and, jointly with Benjamin Monate (CEA) of Paolo Herms, since Oct. 08 (certification of Frama-C/Jessie/Why tool-chain).

C. Paulin is supervising the PhD of Wendi Urribarrí (towards certified libraries) and together with Franck Védrine and Loïc Correnson (CEA-LIST) the PhD of Nicolas Ayache (Verification of System C programs) at CEA LIST to be defended in january 2010.

M. Pouzet is supervising the PhD of Florence Plateau (the theory of N-synchronous systems) to be defended in january 2010. He supervises the PhD of Léonard Gérard since Sept 2008 (a language for N-synchronous systems) and Cédric Auger, started in Sept 2008 (certified compilation of hierarchical state machines).

X. Urbain is (co-)supervising the PhD theses of A. Compaore (with P. Le Gall, on rewriting techniques for (space and time) simulation of biological processes), and of Z. Bouzid (with S. Tixeuil and M. Gradinariu Potop-Butucaru, on models and algorithms for emerging systems).

9.2.2. Supervision of Post-docs and internships

C. Marché supervised the post-doc intern of A. Ayad from Oct 08 to Jan. 09 (behavioral properties of floating-point programs). A. Ayad has now a post-doc position at CEA-List laboratory in Saclay.

C. Marché supervises the post-doc intern of K. Krishnamani since Sep 09 (predicate abstraction techniques for critical C programs).

C. Marché and J.-C. Filliâtre supervised the internship of A. Paskevich in Jul.-Aug. 09 (inductive types in the Why tool). A. Paskevich stayed in our team on an assistant professor position at University Paris-Sud 11.

S. Conchon and É. Contejean supervised the master thesis (Master of Orsay) of M. Iguernelala [39] about the integration of associative and commutative symbols in a decision procedure for linear arithmetic and equality in the *Alt-Ergo* theorem prover.

É. Contejean supervised the post-doc intern of A. Paskevich from Sep 08 to Jan 09 (modelling linear Diophantine equations in *Coq*, for A3PAT). A. Paskevich is now assistant professor at Université Paris-Sud 11, in our team.

S. Conchon and J.-C. Filliâtre supervised the *Ocamlviz* summer project of G. Von Tokarski and J. Robert, funded by Jane Street Capital (NYC, USA).

S. Conchon and J.-C. Filliâtre supervised the undergraduate internship of A. Milchior [41] about the implementation of a new matching algorithm in the *Alt-Ergo* theorem prover.

S. Conchon and É. Contejean supervised the master thesis (Master of Orsay) of M. Iguernelala [39] about the integration of associative and commutative symbols in a decision procedure for linear arithmetic and equality in the *Alt-Ergo* theorem prover.

C. Marché supervised, jointly with S. Boulmé (VERIMAG, Grenoble), the Master's internship of A. Tafat, from Apr. to Sep. 09 (Refinement techniques for object-oriented programs). A. Tafat is now a Phd student in the team.

C. Marché supervised, jointly with A. Giorgetti (LIFC, Besançon), the Master's internship of E. Tushkanova, from Apr. to Sep. 09 (Modular Specification of Java programs). E. Tushkanova is now a PhD student at LIFC under supervision of O. Kouchnarenko.

9.2.3. Graduate courses

- Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/

  In 2009-2010, É. Contejean lectured on advanced rewriting (12h) in the course on "Automated Deduction".

  In 2009-2010, C. Paulin lectured (6h) in the course on "Proof assistants".

  In 2009-2010, M. Pouzet lectured (12h) in the course on "Synchronous Systems".

  In 2009-2010, S. Boldo lectured (10h) in the course on "Foundations of proof assistants".

  In 2009-2010, X. Urbain lectured (12h) in the course on "Automated Deduction".
- M. Pouzet is responsible of a course (24h) on synchronous programming in the Professional Master (ISIC, "Ingénierie des Systèmes Industriels Complexes" of École Polytechnique, University Paris-Sud 11 and INSTN).
- C. Marché lectured at the Winter School on Verification of Object-Oriented Programs (Viinistu, Estonia, 25-29 January 2009) on the *Krakatoa* tool (4h course + 1h practical lab) [40].
- G. Melquiond lectured at the 1st Asian-Pacific Summer School on Formal Methods (Beijing, China, August 24-31, 2009, http://formes.asia/cms/coqschool/2009), one lecture on the SSreflect Coq tactic and one lecture, joint with V. Prevosto (CEA List), on the Why tool.

9.2.4. Other Courses

S. Conchon, L. Mandel, C. Paulin and M. Pouzet are teaching as part of their duty (192h per year) at University Paris-Sud 11. Florence Plateau taught as part of its duty (96h per year) as *"ATER"* at I.U.T Orsay until August.

M. Pouzet is in charge of the Master MISIC on Industrial Systems http://www.dix.polytechnique.fr/chaire-systemes-complexes/ for Paris-Sud 11 (the Master is common to École Polytechnique, INSTN and Paris-Sud 11).

A. Paskevich is teaching as part of his duty (150h during academic year 2009/2010) at IUT d'Orsay, University Paris-Sud 11.

In fall 2009, J.-C. Filliâtre is lecturing (24h) at École Normale Supérieure on programming languages and compilers. In 2008–2009, J.-C. Filliâtre is teaching at École Polytechnique (70h per year).

C. Auger and R. Bardou are teaching as a "moniteur" position (64h per year) at University Paris-Sud 11. This is also the case for F. Bobot and L. Gérard since Sep. 2009.

Since Sep. 2009, A. Tafat, M. Iguernelala are teaching as a "moniteur" position (64h per year) at I.U.T Orsay.

9.3. Industrial Dissemination

G. Melquiond also participates in the meetings of the IEEE-1788 standardization committee on interval arithmetic. The "Technology Transfer and Innovation" INRIA department is funding his travel expenses till late 2011.

Airbus France expressed in 2009 the wish to integrate our tool Alt-Ergo in its process of certification of the critical softwares in their next generation planes. We thus started the procedure of *qualifying* Alt-Ergo in the sense of the DO-178B norm, which fixes the contraints on software development to achieve certification of an avionics software. This is done as part of the new *ADT Alt-Ergo*.

The Frama-C environment has a growing industrial impact, being currently under experimental use and evaluation by U3CAT industrial partners but also other european industrial users, e.g. Fraunhofer FIRST institute (http://www.first.fraunhofer.de/) in Berlin, Germany; Bosch company in Germany; Thalès group, France (within european project ITEA2 SPICES http://www.spices-itea.org).

The Krakatoa/Why chain for Java verification is under use at Gemalto company.

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 prepared an activity related to programs and astronomy for the event *"10e Salon de la culture et des jeux mathématiques"* (May 28-31, 2009). It is based on an OCaml program by J.-C. Filliâtre that gives an accurate graphical representation of the night sky, from any location on Earth, at any date and time. S. Boldo, F. Bobot and R. Bardou animated this activity for children and the general public.

This activity was reused for the event "Fête de la science" organized by the French ministry of Education and Research (November 20-22, 2009). S. Boldo, R. Bardou, S. Conchon, and A. Paskevich animated this activity for children and the general public.

S. Boldo gave a talk for mathematic and technology secondary school teachers. On June 10 th and 16th, the teachers attended several talks in a seminar called "Formation Informatique et Objets Numériques" in order to prepare a computer science option in the secondary schools of the academy of Versailles. A CD was edited by the INRIA with all participants' talks, and the talks are also available on http://www.inria.fr/rocquencourt/ressources/multimedia/formation-informatique-et-objets-numerique/.

S. Boldo gave a talk for mathematic secondary school teachers at their assembly on October 14th.

S. Boldo is invited speaker on December 8th at a special day for a hundred secondary school girls (15-16 years old) to promote women in mathematics and computer science.

S. Boldo was invited to write a popular science article about programmation in DocSciences, a magazine edited by the "Académie de Versailles". This article was then put on the popular science web site interstices [36].

G. Melquiond animated the INRIA stand at the European Research Carrier Fair in Berlin, May 28, 2009.

C. Auger and S. Conchon gave a talk at *"Fête de la science"* on November 23 about the order of magnitude of problems computer science has to deal with. The goal of this talk was to show that naive brute force algorithms can not solve many problems occurring in practice, even with the help of billions of supercomputers. This talk has also been given by S. Conchon during the award ceremony of *"Olympiades de mathématiques"* on May 13, 2009.

10. Bibliography

Major publications by the team in recent years

[1] 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.

[2] J.-L. COLAÇO, B. PAGANO, M. POUZET. *A Conservative Extension of Synchronous Data-flow with State Machines*, in "ACM International Conference on Embedded Software (EMSOFT'05), Jersey city, New Jersey, USA", September 2005.

[3] 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, vol. 4720, Springer, September 2007, p. 148–162.

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

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

[6] J.-C. FILLIÂTRE, P. LETOUZEY. *Functors for Proofs and Programs*, in "Proceedings of The European Symposium on Programming, Barcelona, Spain", Lecture Notes in Computer Science, vol. 2986, April 2004, p. 370–384, http://www.lri.fr/~filliatr/ftp/publis/fpp.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, vol. 3308, Springer, November 2004, 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", vol. 199, n^{o} 1-2, May 2005, 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", vol. 58, n^{o} 1–2, 2004, p. 89–106, http://krakatoa.lri.fr.

Year Publications

Doctoral Dissertations and Habilitation Theses

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

Articles in International Peer-Reviewed Journal

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

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

[14] S. BOLDO, M. DAUMAS, R.-C. LI. *Formally Verified Argument Reduction with a Fused-Multiply-Add*, in "IEEE Transactions on Computers", vol. 58, n^{o} 8, 2009, p. 1139-1145, http://arxiv.org/abs/0708.3722 US .

[15] M. DAUMAS, G. MELQUIOND. *Certification of bounds on expressions involving rounded operators*, in "Transactions on Mathematical Software", vol. 37, n^{o} 1, 2009.

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

Invited Conferences

[17] J.-C. FILLIÂTRE. *Invited tutorial: Why — an intermediate language for deductive program verification*, in "Automated Formal Methods (AFM09), Grenoble, France", H. SAÏDI, N. SHANKAR (editors), 2009.

International Peer-Reviewed Conference/Proceedings

[18] 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.

[19] 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, vol. 5556, Springer, July 2009, p. 91–102.

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

[21] P. CASPI, J.-L. COLAÇO, L. GÉRARD, M. POUZET, P. RAYMOND. *Synchronous Objects with Scheduling Policies: Introducing safe shared memory in Lustre*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), Dublin", June 2009.

[22] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Relaxing Synchronous Composition with Clock Abstraction*, in "Hardware Design using Functional languages (HFL 09), York, UK", march 2009, p. 35-52.

[23] W. EDMONSON, G. MELQUIOND. *IEEE interval standard working group - P1788: current status*, in "Proceedings of the 19th IEEE Symposium on Computer Arithmetic, Portland, OR, USA", J. D. BRUGUERA, M. CORNEA, D. DASSARMA, J. HARRISON (editors), 2009, p. 231–234 US .

[24] C. HURLIN, F. BOBOT, A. J. SUMMERS. *Size Does Matter : Two Certified Abstractions to Disprove Entailment in Intuitionistic and Classical Separation Logic*, in "International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO'09)", July 2009, http://www.lri.fr/~bobot/publis/Hurlin_Bobot_Summers_iwaco09.pdf UK NL .

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

[26] S. LESCUYER, S. CONCHON. *Improving Coq Propositional Reasoning using a Lazy CNF Conversion Scheme*, in "Frontiers of Combining Systems, 7th International Symposium, Proceedings, Trento, Italy", S. GHILARDI, R. SEBASTIANI (editors), Lecture Notes in Computer Science, vol. 5749, Springer, September 2009, p. 287-303.

[27] L. MANDEL, F. PLATEAU. *Abstraction d'horloges dans les systèmes synchrones flot de données*, in "Vingtièmes Journées Francophones des Langages Applicatifs, Saint-Quentin sur Isère", INRIA, January 2009, http://www.lri.fr/~plateau/papers/jfla09.pdf.

[28] M. POUZET, P. RAYMOND. *Modular Static Scheduling of Synchronous Data-flow Networks: An efficient symbolic representation*, in "ACM International Conference on Embedded Software (EMSOFT'09), Grenoble, France", October 2009.

Scientific Books (or Scientific Book chapters)

[29] J.-M. MULLER, N. BRISEBARRE, F. DE DINECHIN, C.-P. JEANNEROD, V. LEFÈVRE, G. MELQUIOND, N. REVOL, D. STEHLÉ, S. TORRES. *Handbook of Floating-Point Arithmetic*, Birkhäuser, 2009.

[30] C. PAULIN-MOHRING. *A constructive denotational semantics for Kahn networks in Coq*, in "From Semantics to Computer Science: Essays in Honor of Gilles Kahn", Y. BERTOT, G. HUET, J.-J. LÉVY, G. PLOTKIN (editors), Cambridge University Press, 2009, http://hal.inria.fr/inria-00431806/.

Research Reports

[31] A. AYAD. *On formal methods for certifying floating-point C programs*, n^{o} RR-6927, INRIA, 2009, http://hal.inria.fr/inria-00383793/, Research Report.

[32] 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.

[33] Y. MOY, C. MARCHÉ. *Jessie Plugin Tutorial, Beryllium version*, INRIA, 2009, http://www.frama-c.cea.fr/jessie.html.

[34] A. PASKEVICH. *Algebraic types and pattern matching in the logical language of the Why verification platform*, n^{o} RR-7128, INRIA, 2009, http://hal.inria.fr/inria-00439232/, Technical report.

[35] E. TUSHKANOVA, A. GIORGETTI, C. MARCHÉ, O. KOUCHNARENKO. *Modular Specification of Java Programs*, n^{o} RR-7097, INRIA, 2009, http://hal.inria.fr/inria-00434452/en/, Technical report.

Scientific Popularization

[36] S. BOLDO. *Demandez le programme!*, February 2009, http://interstices.info/demandez-le-programme, Interstices.

Other Publications

[37] A. AYAD, C. MARCHÉ. *Behavioral Properties of Floating-Point Programs*, 2009, http://hisseo.saclay.inria.fr/ayad09.pdf, Hisseo publications.

[38] R. BARDOU, C. MARCHÉ. *Regions and Permissions for Data Invariants*, 2009, http://www.lri.fr/cepromi/.

[39] M. IGUERNELALA. *Extension modulo Associativité-Commutativité de l'algorithme de clôture par congruence CC(X)*, Université Paris-Sud, 2009, Masters thesis.

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

[41] A. MILCHIOR. *Algorithme de matching, modulo égalité, incrémental, typé et persistant*, 2009.

[42] A. TAFAT, S. BOULMÉ, C. MARCHÉ. *A Refinement Methodology for Object-Oriented Programs*, 2009, http://www.lri.fr/cepromi/.

[43] A. TAFAT. *Invariants et raffinements en présence de partage*, Université Paris 6, 2009, http://www.lri.fr/~marche/tafat09master.pdf, Masters thesis.

[44] W. URRIBARRÍ, C. PAULIN-MOHRING. *Modules and Refinement in Why*, October 2009, http://www.lri.fr/cepromi/.

References in notes

[45] *The MAUDE System*.

[46] 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, vol. 3582, Springer, July 2005, http://jandronick.free.fr/publi/FM2005.pdf.

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

[48] A. BANERJEE, D. A. NAUMANN, S. ROSENBERG. *Regional Logic for Local Reasoning about Global Invariants*, in "European Conference on Object-Oriented Programming (ECOOP), Paphos, Cyprus", July 2008.

[49] M. BARNETT, R. DELINE, M. FÄHNDRICH, K. R. M. LEINO, W. SCHULTE. *Verification of object-oriented programs with invariants*, in "Journal of Object Technology", vol. 3, n^{o} 6, June 2004, p. 27–56.

[50] 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, vol. 1512, Springer, 1998, p. 28-45.

[51] A. BENVENISTE, P. CASPI, S. A. EDWARDS, N. HALBWACHS, P. LE GUERNIC, R. DE SIMONE. *The synchronous languages 12 years later*, in "Proceedings of the IEEE", vol. 91, n^{o} 1, January 2003.

[52] G. BERRY, G. GONTHIER. *The Esterel synchronous programming language, design, semantics, implementation*, in "Science of Computer Programming", vol. 19, n^{o} 2, 1992, p. 87-152.

[53] Y. BERTOT, N. MAGAUD, P. ZIMMERMANN. *A Proof of GMP Square Root*, in "Journal of Automated Reasoning", vol. 29, n^{o} 3-4, 2002, p. 225–252.

[54] D. BIERNACKI, J.-L. COLAÇO, G. HAMON, M. POUZET. *Clock-directed Modular Code Generation of Synchronous Data-flow Languages*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), Tucson, Arizona", June 2008.

[55] D. BIERNACKI, J.-L. COLAÇO, M. POUZET. *Clock-directed Modular Code Generation from Synchronous Block Diagrams*, in "Workshop on Automatic Program Generation for Embedded Systems (APGES 2007), Salzburg, Austria", October 2007, http://www-fp.dcs.st-and.ac.uk/APGES/OnlineProceedings/11-Pouzet.pdf.

[56] F. BOBOT, S. CONCHON, É. CONTEJEAN, S. LESCUYER. *Implementing Polymorphism in SMT solvers*, in "SMT 2008: 6th International Workshop on Satisfiability Modulo", C. BARRETT, L. DE MOURA (editors), ACM International Conference Proceedings Series, vol. 367, 2008, p. 1–5, http://www.lri.fr/~conchon/publis/conchon-smt08.pdf.

[57] 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, vol. 4130, Springer, August 2006, p. 52-66, http://www.lri.fr/~sboldo/files/ijcar06.pdf.

[58] S. BOULMÉ, M.-L. POTET. *Interpreting invariant composition in the B method using the Spec# ownership relation: a way to explain and relax B restrictions*, in "B 2007", J. JULLIAND, O. KOUCHNARENKO (editors), Lecture Notes in Computer Science, vol. 4355, Springer, 2007, "http://www-lsr.imag.fr/Les.Personnes/Marie-Laure.Potet/PUBLI/B07boulmePotet.pdf.

[59] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming, Philadelphia, Pensylvania", May 1996, http://www.lri.fr/~pouzet/bib/icfp96.ps.gz.

[60] P. CASPI, M. POUZET. *A Co-iterative Characterization of Synchronous Stream Functions*, in "Coalgebraic Methods in Computer Science (CMCS'98)", Electronic Notes in Theoretical Computer Science, March 1998.

[61] R. CAVADA, A. CIMATTI, A. FRANZEN, K. KALYANASUNDARAM, M. ROVERI, R. SHYAMASUNDAR. *Computing Predicate Abstractions by Integrating BDDs and SMT Solvers*, in "Formal Methods in Computer Aided Design", 2007, p. 69-76.

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

[63] A. CIMATTI, A. FRANZEN, A. GRIGGIO, K. KALYANASUNDARAM, M. ROVERI. *Tighter Intergration of BDDs and SMT for Predicate Abstraction*, in "Design, Automation & Test in Europe, Dresden. Germany", IEEE, March 2010.

[64] D. G. CLARKE, J. M. POTTER, J. NOBLE. *Ownership Types for Flexible Alias Protection*, in "Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98)", ACM Press, 1998, p. 48–64.

[65] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *Synchronizing Periodic Clocks*, in "ACM International Conference on Embedded Software (EMSOFT'05), Jersey city, New Jersey, USA", September 2005.

[66] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous Kahn Networks: a Relaxed Model of Synchrony for Real-Time Systems*, in "ACM International Conference on Principles of Programming Languages (POPL'06), Charleston, South Carolina, USA", January 2006.

[67] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Abstraction of Clocks in Synchronous Data-flow Systems*, in "The Sixth ASIAN Symposium on Programming Languages and Systems (APLAS), Bangalore, India", December 2008, http://www.lri.fr/~plateau/papers/aplas08.pdf.

[68] J.-L. COLAÇO, G. HAMON, M. POUZET. *Mixing Signals and Modes in Synchronous Data-flow Systems*, in "ACM International Conference on Embedded Software (EMSOFT'06), Seoul, South Korea", October 2006.

[69] J.-L. COLAÇO, M. POUZET. *Clocks as First Class Abstract Types*, in "Third International Conference on Embedded Software (EMSOFT'03), Philadelphia, Pennsylvania, USA", October 2003.

[70] J.-L. COLAÇO, M. POUZET. *Type-based Initialization Analysis of a Synchronous Data-flow Language*, in "International Journal on Software Tools for Technology Transfer (STTT)", vol. 6, n^{o} 3, August 2004, p. 245–255.

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

[72] S. CONCHON, J.-C. FILLIÂTRE. *A Persistent Union-Find Data Structure*, in "ACM SIGPLAN Workshop on ML, Freiburg, Germany", ACM, October 2007, p. 37–45, http://www.lri.fr/~filliatr/ftp/publis/puf-wml07.pdf.

[73] S. CONCHON, J.-C. FILLIÂTRE. *Union-Find Persistant*, in "Dix-huitièmes Journées Francophones des Langages Applicatifs", INRIA, January 2007, p. 135–149, http://www.lri.fr/~filliatr/ftp/publis/puf.pdf.

[74] S. CONCHON, J.-C. FILLIÂTRE. *Semi-Persistent Data Structures*, in "17th European Symposium on Programming (ESOP'08), Budapest, Hungary", April 2008, http://www.lri.fr/~filliatr/ftp/publis/spds-rr.pdf.

[75] S. CONCHON, J.-C. FILLIÂTRE, F. LE FESSANT, J. ROBERT, G. VON TOKARSKI. *Observation temps-réel de programmes Caml*, in "Vingt-et-unièmes Journées Francophones des Langages Applicatifs, Vieux-Port La Ciotat, France", INRIA, January 2010.

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

[77] 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, vol. 3632, Springer, July 2005, p. 7–22.

[78] É. CONTEJEAN, P. COURTIEU, J. FOREST, A. PASKEVICH, O. PONS, X. URBAIN. *A3PAT, an Approach for Certified Automated Termination Proofs*, in "Partial Evaluation and Program Manipulation, Madrid, Spain", ACM Press, january 2010.

[79] 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, vol. 4720, Springer, September 2007, p. 148–162.

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

[81] P. COURTIEU, J. FOREST, X. URBAIN. *Certifying a Termination Criterion Based on Graphs, Without Graphs*, in "21th International Conference on Theorem Proving in Higher Order Logics", S. TAHAR, O. AIT-MOHAMED, C. MUÑOZ (editors), Lecture Notes in Computer Science, Springer, August 2008, p. 183–198.

[82] P. COURTIEU, G. GBEDO, O. PONS. *Improved matrix interpretations*, in "Proceedings of SOFSEM2010, International Conference on Current Trends in Theory and Practice of Computer Science", Lecture Notes in Computer Science, Springer, January 2010.

[83] K. CRARY, D. WALKER, G. MORRISETT. *Typed Memory Management in a Calculus of Capabilities*, in "ACM Symposium on Principles of Programming Languages (POPL)", ACM Press, 1999, p. 262–275.

[84] 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.

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

[86] 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, vol. 2102, Springer, 2001, p. 246–249.

[87] 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, vol. 1103, Springer, July 1996, p. 93–107.

[88] N. HALBWACHS, P. CASPI, P. RAYMOND, D. PILAUD. *The Synchronous Dataflow Programming Language* LUSTRE, in "Proceedings of the IEEE", vol. 79, n^{o} 9, September 1991, p. 1305-1320.

[89] 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, vol. 3116, Springer, July 2004.

[90] Y. KODA, F. RUSKEY. *A Gray Code for the Ideals of a Forest Poset*, in "Journal of Algorithms", n^{o} 15, 1993, p. 324–340.

[91] 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.

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

[93] P. LETOUZEY. *A New Extraction for Coq*, in "TYPES 2002", H. GEUVERS, F. WIEDIJK (editors), Lecture Notes in Computer Science, vol. 2646, Springer, 2003, http://www.lri.fr/~letouzey/download/NewExtraction.ps.gz.

[94] P. LETOUZEY. *Programmation fonctionnelle certifiée: l'extraction de programmes dans l'assistant Coq*, Université Paris-Sud, July 2004, http://www.lri.fr/~letouzey/download/these_letouzey.ps.gz, Thèse de Doctorat.

[95] L. MANDEL, M. POUZET. *ReactiveML, a Reactive Extension to ML*, in "ACM International Conference on Principles and Practice of Declarative Programming (PPDP), Lisboa", July 2005, p. 82–93, http://www.lri.fr/~mandel/papers/MandelPouzet-PPDP-2005.pdf.

[96] 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, http://www.lri.fr/~marche/marche06sefm.ps.

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

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

[99] G. MELQUIOND. *Floating-point arithmetic in the Coq system*, in "Proceedings of the 8th Conference on Real Numbers and Computers, Santiago de Compostela, Spain", 2008, p. 93–102, http://gappa.gforge.inria.fr/.

[100] G. MELQUIOND. *Proving bounds on real-valued functions with computations*, in "Proceedings of the 4th International Joint Conference on Automated Reasoning, Sydney, Australia", A. ARMANDO, P. BAUMGARTNER, G. DOWEK (editors), Lecture Notes in Artificial Intelligence, vol. 5195, 2008, p. 2–17, http://www.lri.fr/~melquion/soft/coq-interval/.

[101] 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, vol. 1833, Springer, July 2000, p. 270–273, http://bibiserv.techfak.uni-bielefeld.de/talp/.

[102] C. PAULIN-MOHRING. *A library for reasoning on randomized algorithms in Coq - Version 2*, Université Paris Sud, December 2007, http://www.lri.fr/~paulin/ALEA/library.pdf, Description of a Coq contribution.

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

[104] M. SOZEAU, N. OURY. *First-Class Type Classes*, in "21th International Conference on Theorem Proving in Higher Order Logics", S. TAHAR, O. AIT-MOHAMED, C. MUÑOZ (editors), Lecture Notes in Computer Science, Springer, August 2008, http://www.lri.fr/~sozeau/research/publications/First-Class_Type_Classes.pdf.

[105] M. SOZEAU. *Subset coercions in Coq*, in "TYPES 2006", T. ALTENKIRCH, C. M. BRIDE (editors), Lecture Notes in Computer Science, vol. 4502, Springer, 2007, p. 237–252, http://www.lri.fr/~sozeau/research/publications/Subset_Coercions_in_Coq.pdf.

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

[107] J.-P. TALPIN, P. JOUVELOT. *Polymorphic Type, Region and Effect Inference*, in "Journal of Functional Programming", vol. 2, n^{o} 3, 1992, p. 245-271.

[108] L. THÉRY. *Proving Pearl: Knuth's algorithm for prime numbers*, in "Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2003)", D. BASIN, B. WOLFF (editors), LNCS, vol. 2758, Springer-Verlag, 2003.

[109] M. TOFTE, J.-P. TALPIN. *Region-based memory management*, in "Information and Computation", vol. 132, n^{o} 2, 1997, p. 109–176.

[110] 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, Thèse de Doctorat.

[111] J. VUILLEMIN. *On Circuits and Numbers*, Digital, Paris Research Laboratory, 1993, Technical report.