INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Proval

Proof of programs

Saclay - Île-de-France

THEME SYM

Activity Report

2008

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, since Nov. 08]

Faculty Member

Christine Paulin-Mohring [Team Leader, Professor University Paris-Sud 11 (on leave as DR INRIA until Aug. 08), HdR]
Marc Pouzet [Professor University Paris-Sud 11, member of IUF, HdR]
Sylvain Conchon [Assistant Professor Université Paris-Sud 11]
Louis Mandel [Assistant Professor Université Paris-Sud 11]

Technical Staff

Aurélien Oudot [Engineer INRIA, until Oct. 08]
Nicolas Stouls [Expert Engineer INRIA, until Sep. 08]

PhD Student

Cédric Auger [MENRT, since Sep. 08]
Romain Bardou [ÉNS Cachan until Aug. 08, then University Paris-Sud 11]
Nicolas Bertaux [CORDI INRIA, since Dec. 08]
François Bobot [ÉNS Cachan, since Sep. 08]
Thierry Hubert [CIFRE, Dassault aviation, Suresnes, until Jun. 08]
Léonard Gérard [ÉNS Lyon, since Sep. 08]
Johannes Kanig [CORDI INRIA]
Stéphane Lescuyer [INRIA, on leave from X-Telecom]
Yannick Moy [CIFRE France Télécom, Lannion]
Florence Plateau [MENRT-ATER University Paris-Sud 11]
Nicolas Rousset [CIFRE Gemalto, Meudon, until Jun. 08]
Matthieu Sozeau [MENRT University Paris-Sud 11-CNRS]
Wendi Urribarrí [France-Venezuela scholarship]

Post-Doctoral Fellow

Yann Régis-Gianas [Post-doc INRIA, until Sep. 08]
Ali Ayad [Post-doc CNRS, since Sep. 08]
Andriy Paskevych [Post-doc CNRS, since Sep. 08]

Administrative Assistant

Valérie Berthou [TR INRIA, until Sep. 08]
Régine Bricquet [TR INRIA, since Sep. 08]

Other

Maxime Beauquier [Master MPRI, Apr. to Aug. 08]
Paul Brunet [Internship ÉNS Lyon, June and July 08]
Steven Gay [Master MPRI, Apr. to Aug. 08]

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

In his thesis [13], Matthieu Sozeau developed two important tools (`Program` and type classes [38]) corresponding to a major improvement in using the *Coq* proof assistant as an environment for correct program construction. They are integrated in the new version of *Coq* V8.2 [46], [47]. These tools are based on mechanisms for automatic generation of verification conditions which greatly improve the input language of the assistant while preserving its safe kernel. These constructions have quickly gained the interest of academic groups all over the world (e.g. G. Morrisset’s group at Harvard University, USA, for the Ynot project, http://www.eecs.harvard.edu/~greg/ynot/).

The Frama-C platform (http://frama-c.cea.fr), developed in cooperation with CEA List, was publicly released in 2008, together with a proposal for a *Behavioral Interface Specification Language* for C code [43]. The Jessie plugin of this platform connects to our Why platform, and thus provides a tool for deductive verification of C source code. This plugin implements new features coming from Yannick Moy's PhD thesis which will be defended in January 2009. This platform is the major deliverable for the CAT ANR project, which has been very positively evaluated. It has been downloaded hundreds of times. We know that this release is under use by several industrial collaborators (Airbus, Dassault Aviation, Hispano-Suiza, etc.) and academic institutions all over the world (e.g. Fraunhofer First, http://www.first.fraunhofer.de/en/home as part of the ITEA2 project ES_PASS).

3. Scientific Foundations

3.1. Higher-Order Functional Languages

Keywords: *Dependent types*, *Floating-point programs*, *Functional programming*, *Higher-order languages*, *Probabilistic programs*, *Type theory*.

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

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 adressed by M. Sozeau who proposes an extension of the *Coq* input language for building programs [99]. 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 developped in colaboration with N. Oury a system of Type Classes in *Coq* [38] which gives overloading in programs and proofs and facilitates the development of generic tactics.

The second problem has been adressed 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 [89], [90], 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 [87].

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 [100]. 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 [62].

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 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

Keywords: *ANSI C*, *Formal specification and verification*, *Java*, *Java Card*, *Verification condition generator*.

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Thierry Hubert, Johannes Kanig, Claude Marché, Yannick Moy, Aurélien Oudot, Christine Paulin-Mohring, Nicolas Rousset, Nicolas Stouls, 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* [80].

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

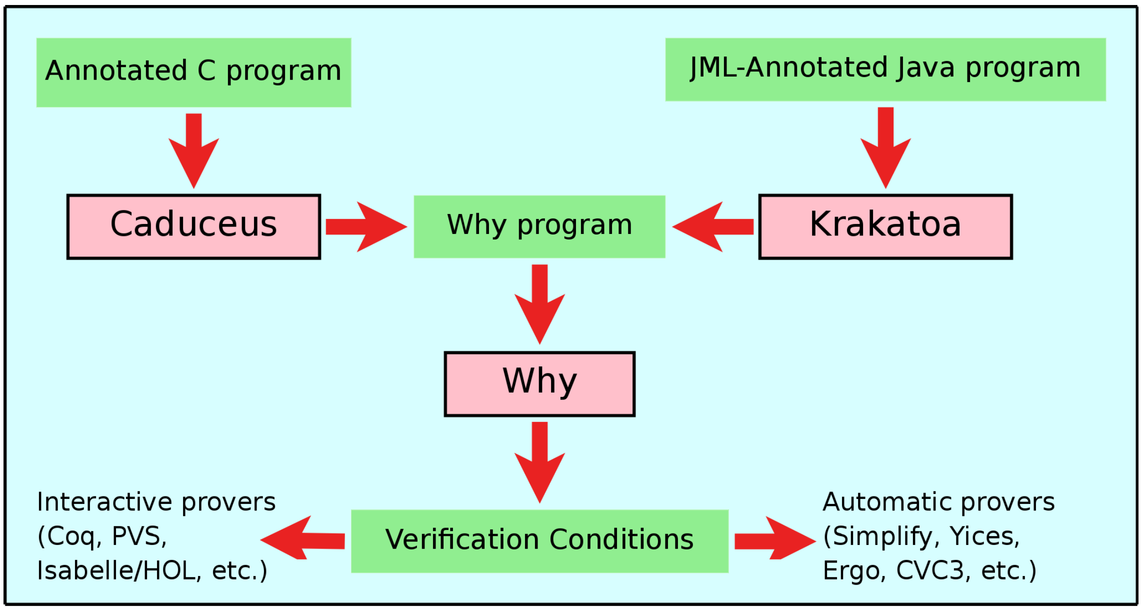
Annotated C program
JML-Annotated Java program
Caduceus
Why program
Krakatoa
Why
Interactive provers (Coq, PVS, Isabelle/HOL, etc.)
Verification Conditions
Automatic provers (Simplify, Yices, Ergo, CVC3, etc.)

Figure 1. Overview of the platform architecture

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.

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 [95]. 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 [86] 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 [66] 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 [55], [93].

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

Keywords: *Combination*, *Decision procedures*, *Proof traces*, *Rewriting*, *Termination*.

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer, Claude Marché.

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* [101], and further generalizations [94]. 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 [96] and then to membership equational programs [79], a paradigm used in the *Maude* system [54].

3.3.2. Decision Procedures

3.3.2.1. Combination

Our research related to combination of decision procedures was initiated by a result [82] 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], [75].

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 [88].

3.3.2.3. The Alt-Ergo theorem prover

It would be more convenient to deal with polymorphism directly in the theorem prover. Unfortunately, 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.

Alt-Ergo is based on $CC(X)$ a generic congruence closure algorithm 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 (5000 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.

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* [76].

É. Contejean and the CNAM participants of the A3PAT project, Pierre Courtieu, Julien Forest, Olivier Pons and Xavier Urbain 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 [77][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 which 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 [56], the main modularity theorem for termination presented in the thesis of Urbain [101] as well as innermost termination, dependency pairs for it and its equivalence with standard termination in some specific cases [83].

The main improvement over the previous approach [76] 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

Keywords: *Compilation*, *Kahn networks*, *Real-time embedded systems*, *Static analysis*, *Synchronous languages*, *Type systems*.

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 [59], [84], [60], [64]. 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] [69], various type-based static analysis [70], [71] and modular compilation into sequential code [65], [61][22].

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 [102]. 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 [67] and [68]. 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 [25].

4. Application Domains

4.1. Panorama

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

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

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

Participants: Évelyne Contejean [contact], Claude Marché.

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 which is already available as CVS sources. 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 they will be distributed under the Cecill-C licence.

5.2. The Why platform

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

Participants: Jean-Christophe Filliâtre [contact], Romain Bardou, Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin-Mohring, Nicolas Rousset.

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: interactives ones (*Coq*, Isabelle/HOL, PVS, HOL-light, Mizar) and automatic ones (Simplify, *Alt-Ergo*, and SMT provers Yices, CVC3, Z3, haRVey, etc.). *Why* has been used directly by external researchers in published verifications of non-trivial algorithms (Efficient square root used in GMP, Knuth's algorithm for prime numbers).

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

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

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

Alt-Ergo is a 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 [98] 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://ergo.lri.fr.

5.4. The Program extension of Coq

Keywords: *Dependently Typed Programming*, *Formal Verification*, *Interactive Theorem Proving*, *Verification Conditions*.

Participant: Matthieu Sozeau [contact].

Program is an extension to *Coq* which eases the construction of dependently-typed programs using an original type system, generating VCs to be solved interactively using the prover. It is distributed under the LGPL licence along with *Coq* (http://coq.inria.fr) as a contribution since 2005. It is used by students and researchers working with dependent types, in the U.S.A. (Harvard, Portland), U.K. (Edinburgh) and France (University Paris 7 and Paris-Sud 11, Polytechnique).

5.5. Lucid Synchrone

Keywords: *Synchronous languages*, *causality analysis*, *compilation*, *type and clock inference*.

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., the 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.

5.6. Reactive ML

Keywords: *Programming language*, *Synchronous reactive programming*, *concurrent systems*, *dedicated type-systems*.

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 of 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 mobile ad hoc networks at the University Paris 6 and for the simulation of sensor networks at France telecom and Verimag (CNRS, Grenoble).

5.7. Bibtex2html

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

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.

5.8. Ocamlgraph

Keywords: *Graph*, *Library*, *Ocaml*.

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.

5.9. Mlpost

Keywords: *Library*, *Ocaml*.

Participants: Jean-Christophe Filliâtre [contact], Johannes Kanig, Stéphane Lescuyer, Romain Bardou.

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 is to be presented at JFLA'09 [57].

5.10. The Gappa tool

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

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.

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. Some tools, e.g. Caduceus and Sollya, also rely on Gappa to compute or prove expression enclosures.

It is distributed as open source, under a Cecill-B / GPL dual-license, at http://lipforge.ens-lyon.fr/www/gappa/. Most of the work on this tool was done while in the Arénaire team (INRIA Rhône-Alpes).

In 2008, a "gappa" tactic was added to the *Coq* proof assistant, so that a user can directly call Gappa from *Coq* in order to prove properties related to the certification of floating-point programs.

5.11. The Interval package for Coq

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

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.msr-inria.inria.fr/~gmelquio/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. Higher-order functional programs

Participants: Sylvie Boldo, Évelyne Contejean, Jean-Christophe Filliâtre, Guillaume Melquiond, Christine Paulin-Mohring, Yann Régis-Gianas, Matthieu Sozeau.

6.1.1. Programming and reasoning with dependent types

M. Sozeau continued work on developing extensions to the higher-order functional programming language which consistitutes the kernel of *Coq*. With N. Oury, he designed an interpretation of the Type Classes concept from Haskell and Isabelle in a dependent type theory [38]. The system has been implemented and documented [47] as part of Coq v8.2. It provides an overloading system in programs and proofs and extends the concept to dependent types which gives rise to some original uses. In particular, a new version of the setoid-rewriting tactic of *Coq* has been constructed on top of these classes [48]. The new tactic is more general and customizable than the previous one. Finally, some new tactics to do dependent elimination have been implemented and various other improvements will be part of *Coq* 8.2 [49].

Y. Régis-Gianas together with François Pottier from the Gallium project developped in his thesis a Hoare logic for call-by-value functional programs which is implemented in the Pangolin system which generates proof obligations from pure functional programs annotated with specification. He presented his work at the MPC'08 conference [37].

6.1.2. Randomized programs

The work of C. Paulin and Ph. Audebaud from ÉNS Lyon for modeling probabilistic programs in *Coq* as probability measures will be published in a special issue of the journal Science of Computer Programming [14]. The corresponding *Coq* library [97] is based on a general theory of ordered sets and cpos. It contains high-level theorems for analysing recursive programs. It is currently used in Verimag (Grenoble) and in the Everest INRIA team (Sophia-Antipolis) as the basis for the CertiCrypt environment for formalizing proofs in computational cryptography in the framework of the SCALP project.

Y. Régis-Gianas started the development of a conservative extension to the Why platform in order to deal with probabilistic programs using the monadic approach of C. Paulin and Ph. Audebaud. As a first step, the system of effects of Why has been extended to detect the non-deterministic subterms of a program and to view them through a particular monadic interpretation. The implementation details of this interpretation are being investigated.

6.1.3. Floating-point algorithms

S. Boldo has been working with G. Melquiond (previously ÉNS Lyon and INRIA-Microsoft Research, now INRIA researcher in our team) on the properties of odd rounding. This has led them to a very original method to compute with correct rounding a FMA (Fused Multiply and Add) or a sum [17].

An established collaboration with M. Daumas (Université of Perpignan Via Domitia) and R.-C. Li (University of Texas at Arlington, USA) has been carried on about argument reductions [16]. This is the first step and a very delicate one to compute elementary functions (exponential, sine...).

G. Melquiond has performed a formalization of a multi-precision multi-radix floating-point arithmetic in the *Coq* proof assistant. The novelty of this approach with respect to existing formalizations is that all the functions are computable by reduction in the system [34]. This opens the possibility to rely on numerical computations inside a formal proof.

6.1.4. A Generic Graph Library for Objective Caml

S. Conchon, J.-C. Filliâtre and J. Signoles designed and implemented Ocamlgraph, a highly generic graph library for the programming language Ocaml. This library features a large set of graph data structures—directed or undirected, with or without labels on vertices and edges, as persistent or mutable data structures, etc.—and a large set of graph algorithms. Algorithms are written independently from graph data structures, which allows combining user data structure (resp. algorithm) with Ocamlgraph algorithm (resp. data structure). Genericity is obtained through massive use of the Ocaml module system and its functions, the so-called functors. This work has been presented at the *Eight Symposium on Trends in Functional Programming* [74] and selected to appear in the book series *Trends in Functional Programming* (Intellect Publishing) [28].

6.1.5. Purely Applicative Data Structures

J.-C. Filliâtre designed an Objective Caml implementation which generalizes *ropes*, a purely applicative data structure for large strings introduced by Boehm, Atkinson and Plass in 1995. In this generalization, functors are used to abstract ropes over the type of character and over the underlying data structure for rope sub-elements. This has many benefits, ranging from extensible arrays to highly-efficient text editors. This work was presented at JFLA'08 [40]. Ropes are binary trees and can be balanced to improve efficiency of most operations. J.-C. Filliâtre proposed a re-balancing implementation based on Garsia-Wachs's algorithm for optimum binary trees. This work was presented at the SIGPLAN Workshop on ML'08 [30].

6.2. Proof of imperative and object-oriented programs

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Thierry Hubert, Johannes Kanig, Claude Marché, Yannick Moy, Aurélien Oudot, Christine Paulin-Mohring, Nicolas Rousset, Nicolas Stouls, Wendi Urribarrí.

6.2.1. Eclipse Plugin for the Why platform

A. Oudot developed a new user interface for the *Why* platform [81], integrated into the *Eclipse* environment, a widely used framework for Java development tools. It is generic, in the sense that it supports both Java and C source code, and it allows to execute in parallel the automatic provers for discharging VCs, with a graphical display of the results. It has been used for teaching at Ecole Polytechnique.

6.2.2. Analysis of C source code

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). This is published as a technical report [43] with significant contributions from C. Marché, Y. Moy and J.-C. Filliâtre.

The *Jessie* intermediate language, first designed in 2007 [92], is now implemented in the *Why* platform, and 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 [45]. The *Jessie* plugin is now to main tool for adding new experimental features coming from new theoretical studies as described below.

6.2.3. Floating-Point C Programs

S. Boldo and J.-C. Filliâtre introduced in [1] annotations specific to floating-point arithmetic that have been successfully applied to several floating-point programs.

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 [15].

S. Boldo studied programs computing matrix transformations with M. Daumas (Université of Perpignan Via Domitia) and P. Giorgi (University of Montpellier 2). The annotations are especially tricky here as the data are matrix [24].

As part of the CerPAN 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 submitted article describes this application and this technique.

Claude Marché and Ali Ayad develop a floating-point extension of the Jessie plugin of the Frama-C platform, based on [1] in order to handle numerical C programs. An extension of the approach to support full floating-point arithmetic, that is involving special values for infinity and Nan (not-a-number), is in progress.

6.2.4. Separation Analysis, avionics case studies

T. Hubert and C. Marché proposed in 2007 a static analysis of pointer programs to discover a partition of memory heap in regions, where it is guaranteed that no pointer can access to several regions [85]. This analysis is accurate thanks to the use of parametric regions: pointers as parameters of functions may have *polymorphic* regions, which allows to call this function with different regions as instances. In 2007, a successful experiment has been fulfilled (100% of VCs proved) on an excerpt of a real embedded program for avionics provided by Dassault Aviation (3kloc). In T. Hubert PhD thesis, defended in June 2008, a detailed and slightly improved separation analysis is presented, and another experiment is reported, on a complete 70kloc C programs, where 97.5 % of VCs are proved. The failure cases are analyzed in Hubert's thesis. The thesis is considered very successful by Dassault Aviation, and collaboration will continue within a new ANR project U3CAT which will start in 2009.

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.

6.2.5. Automatic generation of annotations

In the context of the three PhD of Hubert, Rousset and Moy, all of them aimed at applying deductive verification techniques to industrial programs, a key issue identified was the need of an automatic synthesis of annotations.

In Hubert's PhD [11], an ad-hoc technique is presented which generates most of the needed precondition of safety of memory accesses. A more general and systematic approach has then been studied by Moy and Rousset, 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.

Rousset proposed a combination of forward and backward abstract interpretation, in order to discover preconditions and post-conditions of sub-programs [12]. Moy proposed to combine abstract interpretation, weakest precondition calculus and quantifier elimination. This has been presented at the VMCAI'08 conference [36]. N. Rousset and Y. Moy implemented this new analyses in the *Jessie* tool. This implementation is roughly 3000 lines of Ocaml for the plugin part inside *Jessie*. It calls the external library APRON (http://apron.cri.ensmp.fr/library/) for dealing with abstract domains of octagons or polyhedrons. The resulting automatic annotation generation served as a based for further work below.

6.2.6. Memory Safety Analysis, Non-null by default policy

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 in previous section 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 been proved almost correct (99% of the VCs). This is reported in Y. Moy PhD thesis, which will be defended in January 2009.

C. Marché and Rousset studied memory safety for Java programs in particular, and they proposed a new variant of "non-null policy" which amounts to consider each reference as non-null, unless specified nullable by a user annotation. The novelty is their non-null policy is that non-null information is kept as a static typing information in the Jessie intermediate language, in order to automatically discharge a large part of memory safety VCs. This has been experimented in combination with the automatic generation of annotations, on Javacard applets provided by Gemalto [12]. The conclusion of the experiments is the proposal of a methodology which allows to almost automatically perform safety analysis: first, class invariants together with nullable annotations can be extracted from UML class diagrams, then automatic generation and propagation of annotations can be performed by contextual abstract interpretation, and finally memory safety VCs that have not been discharged by the non-null policy static typing can be sent to provers for validation.

6.2.7. Unions and Casts in C

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.

6.2.8. Structure Invariants and Ownership Systems

One major aspect of the new ARC CeProMi is the support of structure invariants in reasoning on pointer programs. It is one direction of research of R. Bardou PhD thesis. A first proposal is to reuse the Boogie methodology [58] for class invariants in Object Oriented programs, based on an *ownership* relation between objects. One challenge was to combine this approach with the component-as-array memory modeling of *Jessie*, and to generate appropriate verification conditions for structure invariants. A solution is proposed in a paper presented at the FTfJP'08 workshop [21].

6.2.9. Security Properties on C Programs

In the context of the PFC project, Nicolas Stouls started in September 2007 to study existing approaches to express some high level security properties in annotation languages, such as JML for Java programs or ACSL for C programs. He studied a very low level specification of a secured boot loader called OSLO (Open Secure LOader), which includes some routines in assembly language. The code together with specifications is available.

Together with Julien Groslambert from Trusted Labs, N. Stouls proposed a method to generate pre and post condition on methods from a temporal property expressed in LTL or as a Büchi automata. He designed the corresponding plugin for Frama-C called Aorai.

The study of the OSLO example using this approach showed some limitation on the number of proof obligations that were automatically discharged by the system. As a continuation of the work described in [78], N. Stouls together with J.-F. Couchot and A. Giorgetti from the CASSIS projetc proposed a new method for selecting relevant hypotheses. The method is described in a research report [44].

6.2.10. Abstraction and modularity

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 is currently investigating a module system for the input language of the *Why* VCG. She defined notions of modules and interfaces, proposed refinement rules and is currently proving a principle of state variables hiding.

6.2.11. 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 is currently working on an extension of the *Why* tool that can deal with these features in a modular way. On top of this extension one will be able to verify ML programs using a memory model combined to a translation to this extension of *Why*, similarly to what is done in *Caduceus* or *Krakatoa*.

6.3. Automated Deduction

Participants: Maxime Beauquier, Sylvain Conchon, Évelyne Contejean, Thierry Hubert, Stéphane Lescuyer, Claude Marché, Guillaume Melquiond.

6.3.1. Formalization of an efficient SAT-solver

In [31], we have presented a *Coq* formalization of an algorithm deciding the satisfiability of propositional formulas (SAT). This *SAT solver* is described as a set of inference rules (already described in [39]) in a manner that is independent of the actual representation of propositional variables and formulas. We prove soundness and completeness for this system, and instantiate our solver directly on the propositional fragment of *Coq*'s logic in order to obtain a fully reflexive tactic. Such a tactic represents a first and important step towards our ultimate goal of embedding an automated theorem prover inside the *Coq* system. We also extract a certified Ocaml implementation of the algorithm.

6.3.2. 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 [23]. 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 [26].

6.3.3. 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 where only ancestors of the most recent version 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 [27].

6.3.4. Automated proofs and certificates

6.3.4.1. Coccinelle and CiME's traces

The modeling of AC-rewriting of Coccinelle has been used by M. Beauquier [51] during his master thesis to build the beginnings of a new tactic in the *Coq* proof assistant in order to perform automatically rewriting modulo Associativity-Commutativity for first-order terms.

É. Contejean gave a general presentation of Coccinelle at the Types'08 meeting [41], and with Xavier Urbain, a more specific invited talk [42] on the termination aspects of Coccinelle at the Workshop on Certified Termination (WScT'08). It is now possible to produce termination certificates from the trace given not only by CiME but also by an external termination tool, such Approve.

In 2008, CiME\Coccinelle, as well as Approve\Coccinelle were entrants of the termination competition in the category for certified termination proofs.

6.3.4.2. Proofs of bounds on real-valued expressions

G. Melquiond has built a library for automatically proving bounds on expressions in the *Coq* system [35]. 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.

6.4. Synchronous Programming

Participants: Louis Mandel, Florence Plateau, Marc Pouzet.

6.4.1. Tools

L. Mandel improved the development of ReactiveML, an extension of Objective Caml with reactive constructs [91]. During the period, an interactive toplevel has been proposed for the language [33] (with F. Plateau). A reference article is published in [19]. L. Mandel also continued a collaboration with L. Maranget (Moscova Team, INRIA Rocquencourt) with the development of the JoCaml language, which is an extension of OCaml with concurrency, based on the join-calculus [32].

6.4.2. Automatic Distribution

In collaboration with A. Girault and G. Delaval (INRIA Grenoble, Team PopArt), M. Pouzet has worked on the modular distribution of synchronous programs. The goal is to automatically produce a distributed implemention of a synchronous program. The solution is based on a introduction of special annotations (or *locations*). Then, a dedicated type-system is used to check the coherence of annotations with respect to a description of the architecture. This work is published in [29].

6.4.3. 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.

Most of the algebraic properties of this new abstraction have been formalized and proved in the proof assistant *Coq*.

L. Mandel, F. Plateau and M. Pouzet have worked on the relaxed model of synchrony during the period [25].

6.4.4. 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 [22].

7. Contracts and Grants with Industry

7.1. System@tic: PFC

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

The PFC project (Plateforme de Confiance, trusted platforms) is one 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. CIFRE grants

Participants: Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin-Mohring, Nicolas Rousset.

CIFRE, Dassault Aviation We are collaborating with Dassault Aviation in the area of verification of embedded programs written in C.

Thierry Hubert started in February 2005 a thesis on static analysis of functional properties of imperative programs. The goal is to adapt techniques of static analysis for Java or C source code in order to infer appropriate annotations that will be used for formal proofs of the programs. This work is motivated by the need to deal with large applications. The thesis has been defended in June 2008.

CIFRE, Gemalto Nicolas Rousset started his thesis in January 2005. He is working on the conception and validation of secure embedded applications, in particular on smart cards. The idea is to use UML/OCL notations for specifications and to make the link with the tools *Krakatoa* and *Caduceus* as well as Esterel studio. This requires to interpret the semantics of UML diagrams in term of logical formula. The thesis has been defended in June 2008.

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.

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 will last for 3 years.

The goal of the project is to develop an open-source toolkit for analysing industrial-size C programs during development, verification, maintenance and evolution. We address the following issues:

- reusability of components;
- threats detection (division by zero), fault propagation and proof of global properties;
- dependance analysis (control and data flow) for documentation and detection of the impact of modifications.

Partners: CEA, IRISA, Dassault Aviation, Airbus, Siemens.

8.1.2. A3PAT

Participants: Andriy Paskevych, Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer.

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 3 years project funded by ANR, started in December 2005. http://a3pat.ensiie.fr/

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

The coordinator is Xavier Urbain (CNAM). The scientific leaders are Yves Bertot (Inria Sophia), Pierre Casteran (Labri, Bordeaux 1) and É. Contejean (LRI, Orsay). The expected results are

- to define a language well suited for proof traces (equality by rewriting, termination, paramodulation and congruence closure). It is also planed to have certified proofs of local confluence for rewriting systems, hence modelling critical pairs lemma and standard, AC and C unification algorithms is needed
- to implement the theoretical results so as to enable connection between *Coq* and CiME.
- to fully integrate the *Alt-Ergo* theorem prover inside the *Coq* proof assistant using an ad-hoc trace mechanism.

8.1.3. CeProMi

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

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. http://www.lri.fr/cepromi/

The goal is 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 are 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)

8.1.4. 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. http://www-lipn.univ-paris13.fr/CerPAN/

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

8.1.5. FOST

Participants: Sylvie Boldo, Jean-Christophe Filliâtre.

FOST (Formal prOofs of Scientific compuTation programs) is a 3 years ANR "Blanc" project that will shortly begin (Dec 2008 or January 2009). S. Boldo is the coordinator 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.

The partners are: University Paris 13, INRIA Paris - Rocquencourt (Estime).

8.1.6. Hisseo

Participants: Sylvie Boldo, Claude Marché, Guillaume Melquiond, 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.

The partners are: CEA LIST, INRIA Paris - Rocquencourt (Gallium).

8.1.7. 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.

8.1.8. 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.

8.1.9. 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), 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)

8.1.10. SCALP

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

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 Everest, ENS Lyon, LRI, CNAM.

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

8.1.11. PARTOUT

Participants: Louis Mandel, Marc Pouzet.

This project is funded by ANR (program DEFIS).

It will start 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,distributed systems, up to applications deployed on the Web.

8.1.12. DECERT

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer.

DECERT (DEduction and CERTification) is an ANR “Domaines Emergents” project. It will start 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 CEA’s Frama-C tool 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. Coordination Action TYPES

TYPES is a working group in the EU’s 6th framework programme. It started in September 2004 lasted until april 2008. It was a continuation of a number of successful European projects starting in 1997. http://www.cs.chalmers.se/Cs/Research/Logic/Types/

The project involved not less than 33 academic groups in Sweden, Finland, Italy, Portugal, Estonia, Serbia, Germany, France, United Kingdom, Poland ; and industrial research groups at France Telecom and Dassault Aviation.

The aim of the research was to develop the technology of formal reasoning and computer programming based on Type Theory. This is done by improving the languages and computerised tools for reasoning, and by applying the technology in several domains such as analysis of programming languages, certified software, formalisation of mathematics and mathematics education.

8.2.2. 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 academic groups among 15 countries in Belgium, Denmark, Estonia, France, Germany, Ireland, Israel, Italy, The Netherlands, Norway, Poland, 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. Invitations

Dr César Muñoz from the National Institute of Aerospace (NIA, Hampton, Virginia, USA) visited the Proval team for one month, in July 2008, with Digiteo funding support. The cooperation will continue in particular in relation with the Hisseo project, on the subject on analysing avionics code involving floating-point computations.

We had two visits associated with the Orchid project in collaboration with National Taiwan University: Pr. Yih-Kuen Tsay together with his student Ming-Hsien Tsai for one week in July 2008 and Tyng-Ruey Chuang from Academia Sinica for one week in August 2008.

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.

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.

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 2008, she participated to the jury of CR2 hiring committee at INRIA Saclay as well as the national DR2 hiring committee.

M. Pouzet was a member of the *"commission de spécialiste"*, section 27, INPL Nancy, spring 2008.

9.1.3. Collective responsibilities outside INRIA

C. Marché was a member of the *"commission de spécialistes"*, Section 27, University Paris-Sud 11, until Sep. 08.

C. Paulin was a member of the steering committee of the european TYPES working group.

C. Marché is the French National Coordinator for the COST action “Formal Verification of Object-Oriented Programs”.

C. Paulin was an elected member of the board (*“conseil d’administration”*) of University Paris-Sud 11 until july 2008.

É. Contejean was a member of the *“jury de l’agrégation externe de mathématiques”* as an expert in computer science for the hiring session of 2008.

C. Marché is member of the program committee of Digiteo Labs (the world-class research park in Ile-de-France dedicated to information and communication science and technology, http://www.digiteo.fr/)

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

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

S. Conchon was a member of the *“commission de spécialistes”*, Section 27, University Paris-Sud 11, until Sep. 08.

9.1.4. Event organization

C. Paulin together with Philippe Audebaud organised the 9th International Conference on Mathematics of Program Construction MPC’08 in jult 2008. She chaired the programme committee. http://mpc08.lri.fr.

L. Mandel and M. Pouzet organised the 15th International Workshop “SYNCHRON” in 1-5 of December 2008.

9.1.5. Learned societies

M. Pouzet is a member of IFIP Working Group 2.11 (Program Generation) http://www.cs.rice.edu/~taha/wg2.11/.

9.1.6. 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.

9.1.7. Program committees

C. Marché was a member of the program committee of the 7th International Workshop on Rewriting Logic and its Applications (WRLA’08) and is a member of the program committee of the 22th International Conference on Automated Deduction (CADE’09).

C. Paulin was a member of the program committee of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008, Montreal, Quebec, Canada). She was also Program Chair of the 9th International Conference on Mathematics of Program Construction (MPC 2008).

S. Conchon was a member of the program committee of the “Journées Francophones des Langages Applicatifs” (JFLA) 2009 and of the 2008 ACM Sigplan Workshop on ML.

J.-C. Filliâtre was a member of the program committee of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008, Montréal, Québec, Canada) and of the 3rd Automated Formal Methods workshop (AFM08, Princeton, USA).

M. Pouzet was member of the program committe of the “Journées Francophones des Langages Applicatifs” (JFLA) 2008, Real-Time and Network Systems Conference 2008 and AFADL 2008. He is member of the steering committee of the workshop on synchronous programming (SLAP) since 2006.

9.1.8. Invited Presentations

J.-C. Filliâtre was invited speaker at SMT 2008 (Princeton, USA) [20]: *Using SMT solvers for deductive verification of C and Java programs*.

M. Pouzet was invited speaker at ISOR 2008 (Alger, Algeria, 2-6/2008): *Synchrony and Clocks in Khan Process Networks*.

9.1.9. PhD theses defended

Thierry Hubert defended his PhD on June 20th, 2008. He was supervised by C. Marché (CIFRE thesis co-supervised by E .Ledinot, Dassault aviation).

Nicolas Rousset defended his PhD on June 30th, 2008. He was supervised by C. Marché (CIFRE co-supervised by B. Chetali, Gemalto).

Matthieu Sozeau defended his PhD on December 8th, 2008. He was supervised by C. Paulin.

Gwenael Delaval defended his PhD on July 1st, 2008. He was co-supervised by M. Pouzet and A. Girault (INRIA PopArt, Grenoble).

9.1.10. Participation to PhD juries

C. Marché was president of the PhD jury of Olivier Pérès (University Paris-sud 11, September 24th, 2008).

C. Paulin was the president of the PhD jury of Kim Nguyen (University Paris-Sud 11, July 2008), Jérôme Lard (Univ. Paris-Sud 11, Oct. 2008) and Lucie Gentils (Univ. Paris-Sud 11, Sept. 2008). She was a member of the PhD jury of Pierre-Yves Strub (Ecole Polytechnique, Sept. 2008). She reviewed the manuscript and was member of the PhD jury of David Baelde (Ecole Polytechnique, Dec. 2008). She reviewed the manuscript of the HDR of Sandrine Blazy (Univ. Evry, Nov 2008) and was member of the HDR jury of Benjamin Werner (Univ. Paris-Sud 11, Apr. 2008).

M. Pouzet was the president of the PhD jury of Ludovic Samper (INP Grenoble, April 2008), Cécile Hardebolle (Université Paris-Sud 11, Nov. 2008) and Alexandre Chapoutot (Ecole Polytechnique, Dec. 2008). He was a member of the PhD jury of member of the Phd jury of Hugo Venturini (Univ. Joseph Fourier, Grenoble, March 2008), and Ananda Basu (Université Joseph Fourier, Grenoble, Dec. 2008). He reviewed the manuscrit and was member of the PhD jury of Loic Struss (UJF Grenoble, Sept. 2008) and Jean-Vivien Millo (Univ. Nice, Dec. 2008).

9.2. Teaching

9.2.1. Supervision of PhDs and internships

S. Boldo supervised the internship of P. Brunet (ÉNS Lyon student) about the proof of some properties of elementary function computation programs in June and July 2008.

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

S. Conchon and É. Contejean supervised the master thesis (MPRI) of F. Bobot [52] about the integration of non-convex decision procedures in the *Alt-Ergo* theorem prover.

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).

É. Contejean and H. Herbelin supervised the master thesis (MPRI) of M. Beauquier (application of AC-matching to a rewriting tactic modulo AC in the *Coq* proof assistant).

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).

L. Mandel supervised the master thesis (MPRI) of S. Gay [53] about the scope extrusion analisys for ReactiveML (Apr. et Aug. 2008).

L. Mandel and M. Pouzet are supervising the the Ph.D. thesis of Nicolas Bertaux (started December 2008) (parallel execution of reactive programs).

C. Marché is supervising the PhD of Yannick Moy (CIFRE co-supervised by P. Crégut, France Télécom R&D) to be defended in January 2009 and also the PhD of Romain Bardou (ÉNS Cachan).

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.

M. Pouzet was the co-advisor (with Alain Girault) of the PhD. thesis of Gwenaël Delaval (INRIA PopArt Grenoble, start in sept. 2004) which was defended in july. He supervises the PhD of Florence Plateau (the theory of N-synchronous systems), started in Sept 2005. He supervises the PhD of Léonard Gérard (4th year of ENS Lyon) started in Sept 2008 (a language for N-synchronous systems) and Cédric Auger, started in Sept 2008 (certified compilation of hierarchical state machines).

M. Pouzet supervised the master thesis (MPRI) of Léonard Gérard (integer clocks) and the master thesis (M2R Orsay) of Cédric Auger (certified compilation of Mode-automata in Coq).

9.2.2. Graduate courses

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

  In 2008-2009, É. Contejean lectured on advanced rewriting (12h) in the course on “Automated Deduction”.

  In 2008-2009, C. Paulin lectured (12h) in the course on “Proof assistants”.

  In 2007-2008, M. Pouzet lectured (12h) in the course on “Synchronous Systems”.
- Marc Pouzet is responsible of a course (24h) on synchronous programming in the Professional Master (ISIC, “Ingenieurie des Systèmes Industriels Complexes” of École Polytechnique, University Paris-Sud 11 and INSTN).

9.2.3. 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 is teaching as part of its duty (96h per year) as *“ATER”* at I.U.T Orsay.

M. Pouzet is in charge of the second year of the master in Computer Science, for the “professional” branch. He 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 Ecole Polytechnique, INSTN and Paris-Sud 11).

In fall 2008, J.-C. Filliâtre lectured (24h) at École Normale Supérieure on programming languages and compilers. In 2007–2008 and 2008–2009, J.-C. Filliâtre was teaching at École Polytechnique (70h per year).

9.3. Industrial Dissemination

C. Marché presented the techniques of deductive verification for checking safety properties at the “séminaire de réflexion-programmes du CEA-LIST”, Dreux, France, March 25th, 2008.

C. Marché presented our deductive verification approach for C programs at the “Workshop Airbus/partners on formal verification tools strategy”, Toulouse, France, December 4-5, 2008.

S. Boldo and G. Melquiond have participated in the revision of the IEEE-754 (1985) standard on floating-point arithmetic. The new 2008 standard has been approved in June and it is the official standard since August 2008. Both names appear in the author list; An article by S. Boldo appears in the bibliography [63].

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.

9.4. Popularization

F. Plateau and Y. Moy prepared an activity related to logics for the event *“Fête de la science”* organized by the French ministry of Education and Research (November 17-23, 2008). F. Plateau, Y. Moy, J. Kanig, C. Auger, F. Bobot and A. Paskevich animated this activity. One hundred children attended their activity.

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.

S. Boldo gave a talk at "Unithé ou café" on January 2008 to popularize issues related to floating-point arithmetic flaws to all the INRIA Saclay - Île-de-France staff.

S. Boldo was interviewed in March 2008 by J. Jongwane for a podcast on the)i(web site: http://interstices.info/a-propos-calcul-ordinateurs.

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

S. Boldo was invited to write a popular science article about programmation in DocSciences, a magazine edited by the "Académie de Versailles". This special issue of November 2008 is co-edited with the INRIA and is about the basics of computer science [50].

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.

[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.

[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.

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

[12] N. ROUSSET. *Automatisation de la Spécification et de la Vérification d'applications Java Card*, Thèse de Doctorat, Université Paris-Sud, June 2008, http://www.lri.fr/~marche/rousset2008these.pdf.

[13] M. SOZEAU. *Un environnement pour la programmation avec types dépendants*, Thèse de Doctorat, Université Paris-Sud, December 2008, http://mattam.org/research/publications/thesis-sozeau.pdf.

Articles in International Peer-Reviewed Journal

[14] P. AUDEBAUD, C. PAULIN-MOHRING. *Proofs of Randomized Algorithms in Coq*, in "Science of Computer Programming", to appear, 2008, http://www.lri.fr/~paulin/ALEA/random-scp.pdf.

[15] S. BOLDO. *Kahan's algorithm for a correct discriminant computation at last formally proven*, in "IEEE Transactions on Computers", September 2008, http://hal.inria.fr/inria-00171497/.

[16] S. BOLDO, M. DAUMAS, R.-C. LI. *Formally Verified Argument Reduction with a Fused-Multiply-Add*, in "IEEE Transactions on Computers", September 2008, http://arxiv.org/abs/0708.3722.

[17] S. BOLDO, G. MELQUIOND. *Emulation of FMA and Correctly-Rounded Sums: Proved Algorithms Using Rounding to Odd*, in "IEEE Transactions on Computers", vol. 57, n^O 4, 2008, p. 462–471, http://hal.inria.fr/docs/00/08/04/27/PDF/odd-rounding.pdf.

[18] F. DURÁN, S. LUCAS, J. MESEGUER, C. MARCHÉ, X. URBAIN. *Proving Operational Termination of Membership Equational Programs*, in "Higher-Order and Symbolic Computation", vol. 21, n^O 1–2, 2008, p. 59–88, http://www.lri.fr/~marche/duran08hosc.pdf.

Articles in National Peer-Reviewed Journal

[19] L. MANDEL, M. POUZET. *ReactiveML : un langage fonctionnel pour la programmation réactive*, in "Technique et Science Informatiques (TSI)", vol. 27, n^O 9–10/2008, 2008, p. 1097–1128, http://www.lri.fr/~mandel/papers/MandelPouzet-TSI-2007.pdf.

Invited Conferences

[20] J.-C. FILLIÂTRE. *Using SMT solvers for deductive verification of C and Java programs*, in "SMT 2008: 6th International Workshop on Satisfiability Modulo, Princeton, USA", C. BARRETT, L. DE MOURA (editors), 2008.

International Peer-Reviewed Conference/Proceedings

[21] R. BARDOU. *Ownership, Pointer Arithmetic and Memory Separation*, in "Formal Techniques for Java-like Programs (FTfJP'08), Paphos, Cyprus", July 2008, http://romain.bardou.fr/papers/jcown.pdf.

[22] 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.

[23] 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), 2008, http://www.lri.fr/~conchon/publis/conchon-smt08.pdf.

[24] S. BOLDO, M. DAUMAS, P. GIORGI. *Formal proof for delayed finite field arithmetic using floating point operators*, in "Proceedings of the 8th Conference on Real Numbers and Computers, Santiago de Compostela, Spain", M. DAUMAS, J. BRUGUERA (editors), July 2008, p. 113–122, http://hal.archives-ouvertes.fr/docs/00/27/89/89/PDF/rnc.pdf.

[25] 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.

[26] 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.

[27] 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.

[28] S. CONCHON, J.-C. FILLIÂTRE, J. SIGNOLES. *Designing a Generic Graph Library using ML Functors*, in "Trends in Functional Programming", M. T. MORAZÁN (editor), vol. 8, Intellect, 2008, http://www.lri.fr/~filliatr/ftp/publis/ocamlgraph-tfp-8.pdf.

[29] G. DELAVAL, A. GIRAULT, M. POUZET. *A Type System for the Automatic Distribution of Higher-order Synchronous Dataflow Programs*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), Tucson, Arizona", June 2008.

[30] J.-C. FILLIÂTRE. *A Functional Implementation of the Garsia–Wachs Algorithm*, in "ACM SIGPLAN Workshop on ML, Victoria, British Columbia, Canada", ACM, September 2008, http://www.lri.fr/~filliatr/publis/gw-wml08.pdf.

[31] S. LESCUYER, S. CONCHON. *A Reflexive Formalization of a SAT Solver in Coq*, in "Emerging Trends of the 21st International Conference on Theorem Proving in Higher Order Logics", 2008.

[32] L. MANDEL, L. MARANGET. *Programming in JoCaml (Tool Demonstration)*, in "17th European Symposium on Programming (ESOP'08), Budapest, Hungary", April 2008, p. 108–111, http://www.lri.fr/~mandel/papers/MandelMaranget-ESOP-2008.pdf.

[33] L. MANDEL, F. PLATEAU. *Interactive Programming of Reactive Systems*, in "Proceedings of Model-driven High-level Programming of Embedded Systems (SLA++P'08), Budapest, Hungary", Electronic Notes in Computer Science, Elsevier Science Publishers, April 2008, p. 44–59, http://www.lri.fr/~mandel/papers/MandelPlateau-SLAP-2008.pdf.

[34] 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://www.msr-inria.inria.fr/~gmelquio/doc/08-rnc8-article.pdf.

[35] 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.msr-inria.inria.fr/~gmelquio/doc/08-ijcar-article.pdf.

[36] Y. MOY. *Sufficient Preconditions for Modular Assertion Checking*, in "9th International Conference on Verification, Model Checking, and Abstract Interpretation, San Francisco, California, USA", F. LOGOZZO, D. PELED, L. ZUCK (editors), Lecture Notes in Computer Science, vol. 4905, Springer, January 2008, p. 188–202, http://www.lri.fr/~moy/publis/moy08vmcai.pdf.

[37] Y. RÉGIS-GIANAS, F. POTTIER. *A Hoare Logic for Call-by-Value Functional Programs*, in "Mathematics of Program Construction, MPC 2008, Marseille, France", P. AUDEBAUD, C. PAULIN-MOHRING (editors), Lecture Notes in Computer Science, vol. 5133, Springer, July 2008, p. 305–335, http://cristal.inria.fr/~fpottier/publis/regis-gianas-pottier-hoarefp.ps.gz.

[38] 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://mattam.org/research/publications/First-Class_Type_Classes.pdf.

National Peer-Reviewed Conference/Proceedings

[39] S. CONCHON, J. KANIG, S. LESCUYER. *SAT-MICRO : petit mais costaud !*, in "Dix-neuvièmes Journées Francophones des Langages Applicatifs, Étretat, France", INRIA, January 2008, http://www.lri.fr/~conchon/publis/conchon-jfla08.ps.

[40] J.-C. FILLIÂTRE. *Gagner en passant à la corde*, in "Dix-neuvièmes Journées Francophones des Langages Applicatifs, Étretat, France", INRIA, January 2008, http://www.lri.fr/~filliatr/ftp/publis/cordes.pdf.

Workshops without Proceedings

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

[42] É. CONTEJEAN, X. URBAIN. *The A3PAT approach*, in "Workshop on Certified Termination WScT08, Leipzig, Germany", may 2008.

Research Reports

[43] P. BAUDIN, J.-C. FILLIÂTRE, C. MARCHÉ, B. MONATE, Y. MOY, V. PREVOSTO. *ACSL: ANSI/ISO C Specification Language*, 2008, http://frama-c.cea.fr/acsl.html.

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

[45] Y. MOY, C. MARCHÉ. *Jessie Plugin Tutorial, Lithium version*, INRIA, 2008, http://www.frama-c.cea.fr/jessie/main.pdf.

[46] M. SOZEAU. *Program*, Chapter of The Coq Proof Assistant Reference Manual – Version V8.2, 2008, http://coq.inria.fr/.

[47] M. SOZEAU. *Type Classes*, Chapter of The Coq Proof Assistant Reference Manual – Version V8.2, 2008, http://coq.inria.fr/.

[48] M. SOZEAU. *User defined equalities and relations*, Chapter of The Coq Proof Assistant Reference Manual – Version V8.2, 2008.

[49] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual – Version V8.2*, 2008, http://coq.inria.fr.

Scientific Popularization

[50] S. BOLDO. *Demandez le programme!*, in "DocSciences", vol. 5, C.R.D.P. de l'académie de Versailles, November 2008, p. 26–33, http://www.docsciences.fr/-DocSciences-no5-.

Other Publications

[51] M. BEAUQUIER. *Application du filtrage modulo associativité et commutativité (AC) à la réécriture de sous-termes modulo AC dans Coq*, Masters thesis, Master Parisien de Recherche en Informatique, 2008.

[52] F. BOBOT. *Satisfiabilité de formules closes modulo une théorie avec égalité et prédicats*, Masters thesis, Master Parisien de Recherche en Informatique, 2008.

[53] S. GAY. *Analyse d'échappement de portée en ReactiveML*, Masters thesis, Master Parisien de Recherche en Informatique, 2008.

References in notes

[54] *The MAUDE System*, http://maude.cs.uiuc.edu/.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] S. BOLDO, J.-M. MULLER. *Some Functions Computable with a Fused-mac*, in "Proceedings of the 17th Symposium on Computer Arithmetic, Cape Cod, USA", P. MONTUSCHI, E. SCHWARZ (editors), 2005, p. 52-58.

[64] 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.

[65] 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, Extended version available as a VERIMAG tech. report no. 97–07 at www.lri.fr/$\sim$pouzet, March 1998.

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

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

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[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.

[73] S. CONCHON, J.-C. FILLIÂTRE. *Union-Find Persistant*, in "Dix-huitièmes Journées Francophones des Langages Applicatifs", INRIA, January 2007, p. 135–149.

[74] S. CONCHON, J.-C. FILLIÂTRE, J. SIGNOLES. *Designing a Generic Graph Library using ML Functors*, in "The Eighth Symposium on Trends in Functional Programming, New York, USA", M. T. MORAZÁN, H. NILSSON (editors), vol. TR-SHU-CS-2007-04-1, Seton Hall University, April 2007, p. XII/1–13, http://www.lri.fr/~filliatr/ftp/publis/ocamlgraph-tfp07.ps.

[75] S. CONCHON, S. KRSTIĆ. *Strategies for Combining Decision Procedures*, in "Theoretical Computer Science", Special Issue of TCS dedicated to a refereed selection of papers presented at TACAS'03, vol. 354, n^{o} 2, 2006, p. 187–210.

[76] 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.

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

[78] J.-F. COUCHOT, T. HUBERT. *A Graph-based Strategy for the Selection of Hypotheses*, in "FTP 2007 - International Workshop on First-Order Theorem Proving, Liverpool, UK", September 2007.

[79] 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.

[80] J.-C. FILLIÂTRE. *Formal Proof of a Program: Find*, in "Science of Computer Programming", vol. 64, 2006, p. 332–240, http://dx.doi.org/10.1016/j.scico.2006.10.002.

[81] J.-C. FILLIÂTRE, C. MARCHÉ. *The Why/Krakatoa/Caduceus Platform for Deductive Program Verification*, in "19th International Conference on Computer Aided Verification, Berlin, Germany", W. DAMM, H. HERMANNS (editors), Lecture Notes in Computer Science, vol. 4590, Springer, July 2007, p. 173–177.

[82] 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.

[83] 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.

[84] 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.

[85] T. HUBERT, C. MARCHÉ. *Separation Analysis for Deductive Verification*, in "Heap Analysis and Verification (HAV'07), Braga, Portugal", March 2007, p. 81–93.

[86] 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.

[87] 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.

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

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

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

[91] 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.

[92] C. MARCHÉ. *Jessie: an intermediate Language for Java and C Verification*, in "PLPV '07: Proceedings of the 2007 workshop on Programming Languages meets Program Verification, Freiburg, Germany", ACM, 2007, p. 1–2, http://doi.acm.org/10.1145/1292597.1292602.

[93] 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.

[94] 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.

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

[96] 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.

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

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

[99] 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.

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

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

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