INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Proval

Proof of programs

Futurs

THEME SYM

Table of contents

1. Team

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

Team Leader
Christine Paulin [DR INRIA since Sep. 06, on leave from University Paris-Sud 11, HdR]

Team Vice-Leader
Claude Marché [DR INRIA, HdR]

Administrative assistant
Stéphanie Meunier [TR INRIA, until Sep. 07]
Valérie Berthou [TR INRIA, since Sep. 07]

Technical assistant
Aurélien Oudot [Engineer INRIA]
Nicolas Stouls [Expert Engineer INRIA, since Sep. 07]

Research Scientists
Marc Pouzet [Professor University Paris-Sud 11, member of IUF since Sep. 2007, HdR]
Sylvie Boldo [CR INRIA]
Sylvain Conchon [Assistant Professor Université Paris-Sud 11]
Évelyne Contejean [CR CNRS]
Jean-Christophe Filliâtre [CR CNRS]
Louis Mandel [Assistant Professor Université Paris-Sud 11, since Sep. 07]

Ph. D. students
Romain Bardou [ÉNS Cachan, since Sep. 07]
Alexandre Bertails [MENRT University Paris-Sud 11, until Sep. 07]
Thierry Hubert [CIFRE, Dassault aviation, Suresnes]
Johannes Kanig [CORDI INRIA, since Sep. 07]
Stéphane Lescuyer [INRIA, on leave from X-Telecom, since Sep. 07]
Yannick Moy [CIFRE France Télécom, Lannion]
Florence Plateau [MENRT University Paris-Sud 11]
Nicolas Rousset [CIFRE Gemalto, Meudon]
Matthieu Sozeau [MENRT University Paris-Sud 11]
Wendi Urribarrí [France-Venezuela scholarship]

Post-doctoral fellows
Malgorzata Biernacka [Post-doc CNRS, Sep. 06 to Aug. 07]
Dariusz Biernacki [Post-doc INRIA, Sep. 06 to Aug. 07]
Jean-François Couchot [Post-doc INRIA, Sep. 06 to Aug. 07]
Yann Régis-Gianas [Post-doc INRIA, since Dec. 07]

Student interns
Romain Bardou [ÉNS Cachan, Feb. 07 to Aug. 07]
Johannes Kanig [2nd year Master Univ. Dresden, Germany, Jan-Feb. 07, then Jun-Aug 07]
Alexandre Viel [ÉNS Paris, Apr. 07 to Aug. 07]

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 challenge "to be capable of producing reliable software".

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.

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

Matthieu Sozeau and Yannick Moy are PhD students of our team. Each of them has written a single-authored paper that has been accepted for presentation to major conferences: respectively the *International Conference on Functional Programming* [44], and the *International Conference on Verification, Model Checking and Abstract Interpretation* [43].

Since november 2007, Esterel-Technologies distributes the new SCADE 6 design tool for safety-critical system. SCADE 6 is a major change of SCADE and provides new features, largely derived from research work by Marc Pouzet: mix of data-flow and hierarchical state machines as introduced in [63], various dedicated type systems such as a clock calculus [64] or initialization analysis [65] and follows the clock based compilation method of the Lucid Synchrone compiler.

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: Dariusz Biernacki, Sylvie Boldo, Évelyne Contejean, Jean-Christophe Filliâtre, Christine Paulin, Malgorzata Biernacka, 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, which can express logical properties of programs and 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 [45]. The solution is similar to the mechanism subset types and Type Checking Conditions in PVS but a Coq proof term is built and is checked by the Coq kernel.

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). Extraction generates programs by erasing logical parts in constructive proofs of specifications. Such ML programs are then correct by construction. During his PhD thesis, P. Letouzey designed and implemented a new extraction mechanism for the Coq system [77], [78], much more powerful than the old version. With this new extraction, J.-C. Filliâtre and P. Letouzey could verify Ocaml finite sets libraries based on balanced trees [4]. This study showed a bug in the Ocaml implementation: trees could possibly become unbalanced.

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 [75].

Another question is how to reason and compute efficiently inside Coq on proof terms representing annotated programs. This is currently under investigation in collaboration with the LogiCal team as a proposal of an extension of the conversion rule in Coq which ignores the proof terms corresponding to correctness arguments.

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 the probability that a program satisfies a given property.

These are detailed in Section 6.1.2.

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 [84]. 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 [1].

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..." [23].

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 are also starting 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, Sylvie Boldo, Jean-François Couchot, Jean-Christophe Filliâtre, Thierry Hubert, Claude Marché, Yannick Moy, Aurélien Oudot, Christine Paulin, 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 [69].

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* [6].

3.2.1. The Why platform

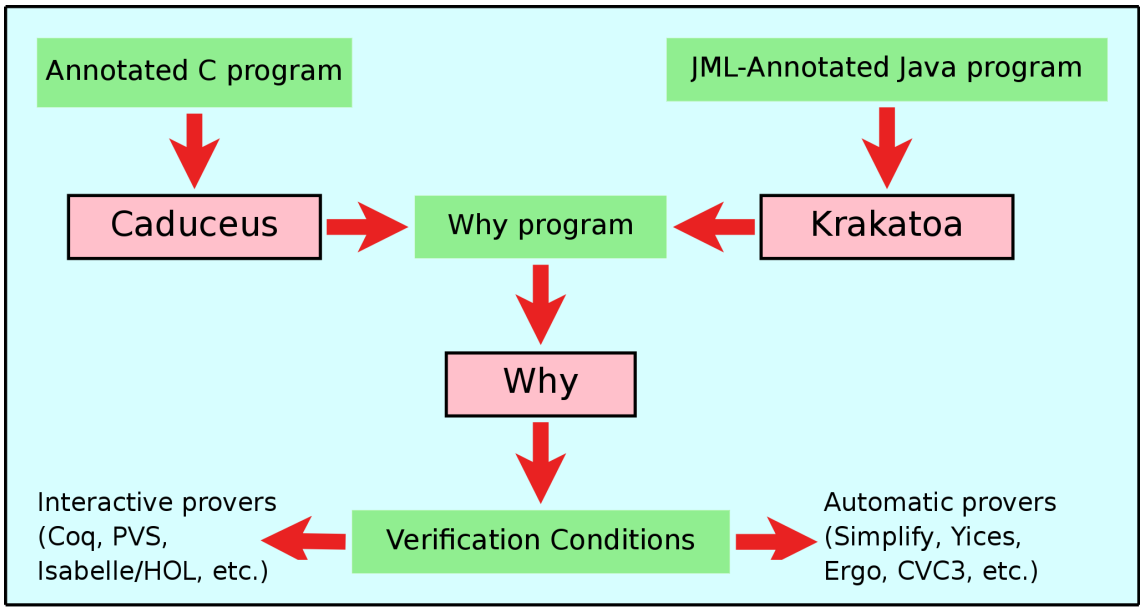

Figure 1. Overview of the platform architecture

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, 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 [80]. 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 [6].

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 will be 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 [74] 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 [59] developed by the company Trusted Logic for experimental purpose. Other Java Card case studies have been conducted in collaboration with Gemalto by J. Andronick and N. Rousset, in particular on global properties and Java Card transactions [53], [81].

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 [7], 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 are currently 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, Jean-François Couchot, Johannes Kanig, 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 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* [85], and further generalizations [9]. These criteria and methods have been implemented into the CiME2 rewrite toolbox [3]. 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 [82] and then to membership equational programs [68], a paradigm used in the *Maude* system [60].

3.3.2. Decision Procedures

3.3.2.1. Combination

Our research related to combination of decision procedures was initiated by a result [70] obtained in collaboration with Shankar's group at SRI-international who develops the PVS environment, showing how decision procedures for disjoint theories can be combined as soon as each of them provides a so-called "canonizer" and a "solver". Existing combination methods in the literature are generally not very well

understood, and S. Conchon had a major contribution, in collaboration with Sava Krstić from OGI School of Science and Engineering (Oregon Health and Science University, USA), which is a uniform description of combination of decision procedures, by means of a system of inference rules, clearly distinguished from their strategy of application, allowing much clearer proofs of soundness and completeness [8], [66].

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 [76].

3.3.2.3. The 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 Ergo which is dedicated to the resolution of polymorphic and multi-sorted proof obligations and takes as input the *Why* syntax.

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.

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

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 (3000 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 [67].

É. Contejean is currently developing a new version of the CiME tool associated with a Coq library called Coccinelle. A new trace generator will output a trace for Coq in the unified framework provided by the Coccinelle library. 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. The main improvement over the previous approach [67] 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.

3.4. Synchronous Programming

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

Participants: Alexandre Bertails, Dariusz Biernacki, 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 [56], [72], [57], [58]. 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] [63], various type-based static analysis [64], [65] and modular compilation into sequential code [73][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 [86]. 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 [61] and [62]. Florence Plateau works since that time on this subject.

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 (http://bibiserv.techfak.uni-bielefeld.de/talp/) 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 (http://www.loria.fr/equipes/protheo/SOFTWARES/CARIBOO/) 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. 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], Jean-François Couchot, Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin, 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, 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. It 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, an Eclipse plugin for the platform has been developed.

5.3. The Ergo theorem prover

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

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

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 Ergo fully supports first order polymorphic logic with quantifiers. Ergo also supports the standard [83] 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. 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/.

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, under the GPL licence, at http://www.lri.fr/~filliatr/bibtex2html/. A Google search of the appropriate banner string reveals that (on November 2007) around 45000 web pages have been generated using Bibtex2html.

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

5.8. 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.

6. New Results

6.1. Higher-order functional programs

Participants: Dariusz Biernacki, Sylvie Boldo, Évelyne Contejean, Jean-Christophe Filliâtre, Christine Paulin, Malgorzata Biernacka, Matthieu Sozeau.

6.1.1. Programming with dependent types

M. Sozeau continued his work on a specific language for writing programs with dependent types in the Calculus of Inductive Constructions, the core language of the Coq proof assistant.

Previous work was based on an extension of conversion to subset types. He extended this approach to general dependent families. He demonstrated the usefulness of this method by developing a non-trivial example (a certified implementation of finger-trees) which has been presented at the ICFP conference [44].

This high-level language has been part of the Coq distribution since 2005. The implementation in Coq (4,000 lines of Ocaml code) gained robustness and completeness. It has been used and stress-tested by a handful researchers around the world.

M. Sozeau is now looking for solutions to implement the extensions needed in Coq's conversion rule to have a faithful translation from the high-level language to the kernel language and ease reasoning about programs resulting from this translation. The main extension, called proof-irrelevance, would also add mathematical power and a more classical feel to the system. This has already been studied by B. Werner [87]. Hence this is joint work with him and the LogiCal team.

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 [11].

A new version of the corresponding Coq library [50] has been developed. It 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) for formalizing proofs in computational cryptography in the framework of the SCALP project.

6.1.3. Floating-point programs

S. Boldo has been working with G. Melquiond (previously ÉNS Lyon and now INRIA-Microsoft Research, Orsay) 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 [14].

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 [51].

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. This is the first step and a very delicate one to compute elementary functions (exponential, sine...). This work has been accepted for publication in the IEEE Transactions on Computers [13].

S. Boldo and A. Viel have been working on floating-point summation programs. After proving a result on the naive summation, they have been working on a compensated summation algorithm by Kahan. This 1965 tricky algorithm is known to have a very good error bound.

S. Boldo and M. Daumas are writing a book on floating-point arithmetic where statements are formally proved.

6.1.4. Theoretical studies of functional languages

In collaboration with Olivier Danvy and Kevin Millikin from the University of Aarhus, D. Biernacki has conducted a continuation of his PhD thesis on theoretical and practical aspects of functional languages with delimited continuations (used in, e.g., non-deterministic programming, partial evaluation, mobile code, Web interactions). In particular, they have carried out further investigation of operational semantics, program transformations and applications of dynamic continuations and their hierarchies, leading to a refined higher-level specification (continuation semantics and CPS transformation), and implementations of dynamic continuations allowing for their incorporation in existing languages. This work has been accepted for publication in the TOPLAS journal [12].

Malgorzata Biernacka has been working on operational aspects of functional programming languages with the goal of conducting formal verification of some of its aspects in the Coq system, using it as a tool for representing languages and their semantics and proving their properties. In particular, a proof of existence of weak head normal forms for a typed language (system T) has been carried out and the extraction mechanism of Coq has produced a program realizing this proof, which is in effect an evaluator for the language. On the one hand, the Coq system provides all the necessary ingredients to formally capture the entire proof in all

detail. On the other hand, the extracted program, though fully functional, is unsatisfactory when it comes to efficiency and type safety, which suggests some possible further fine-tuning of the Coq extraction mechanism.

Another line of work in verification of programming languages is formalization of derivations of abstract machines from high-level operational definitions via reduction semantics. Since both reduction semantics and abstract machines are used at various stages of language design, implementation and practical use, it is desirable to be able to formally state the equivalence (in a certain sense) of all such semantic descriptions, and to be able to prove this equivalence. As a first step towards an automated derivation, M. Biernacka and D. Biernacki have proved correct a method of refocusing that leads from a reduction semantics satisfying standard properties to an abstract machine (this is work in progress; a preliminary version was presented at the 7th International Workshop on Reduction Strategies in Rewriting and Programming [21]). This approach is essentially untyped, allowing reasoning about potentially nonterminating functions. Although Coq imposes termination on recursive definitions, this method can still be represented inside the Coq system, if we use relations rather than functions. The refocusing method works both for pure languages and for languages with computational effects.

6.1.5. 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* [29] and selected to appear in the book series *Trends in Functional Programming* (Intellect Publishing) [16].

6.2. Proof of imperative and object-oriented programs

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

6.2.1. Eclipse Plugin for the Why platform

A. Oudot started in October 2006 the development of a new user interface for the *Why* platform [35], 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. The objective is to offer an easier access to our verification tools, but also advanced features to developers. This plugin has been presented during an ANR-RNTL CAT meeting and has been demonstrated during the "journée INRIA-Industrie Confiance et Sécurité" in October 2007.

6.2.2. Floating-Point C Programs

Sylvie Boldo and Jean-Christophe Filliâtre introduced annotations specific to floating-point arithmetic that have been successfully applied to several short floating-point programs [23].

This methodology is currently applied to a real-life program computing the gradient as part of the CerPAN project. We have 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 upcoming article will describe this techniques and these two applications.

6.2.3. Jessie

For the future development of analyses in the *Why* platform, a new intermediate language called *Jessie* has been designed. It is the target language of translation from C and Java, and is itself translated in a second stage into the input language of the *Why* VCG. The design of *Jessie* has been presented in an invited talk at the PLPV workshop [38].

Y. Moy has developed a plugin to connect *Jessie* to the *Frama-C* platform.

6.2.4. Separation Analysis

T. Hubert and C. Marché made many improvements in the separation analysis whose design started in October 2005. A major improvement is the parametricity of memory variables: briefly speaking, pointers as parameters of functions may have *polymorphic* regions, which allows to call this function with different regions as instances. This has been implemented in *Caduceus*, and applied to a real embedded program for avionics provided by Dassault Aviation (30kloc), and this model improved drastically (from 82% to 99.9%) the number of VCs solved automatically. An article describing this approach was presented at the HAV'07 Workshop [36]. The ten remaining VCs were first proved using the Coq proof assistant, but now are also solved automatically thanks to another technique ; see Section 6.3.2.4.

6.2.5. Memory Safety Analysis

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. A first approach, combining forward and backward abstract interpretation, has been presented at the HAV'07 workshop [41], [49]. A second approach has been proposed, combining abstract interpretation, weakest precondition calculus and quantifier elimination. This second analysis has been accepted for presentation at the VMCAI'08 conference [43].

N. Rousset and Y. Moy implemented this new analysis 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.

6.2.6. 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 has been presented at the C/C++ Workshop [42] and implemented in a new *Jessie* plugin of the *Frama-C* platform, developed in the ANR CAT project.

6.2.7. Invariants and Ownership

R. Bardou did his 2nd year master internship on the subject of structure invariants in the *Jessie* intermediate language. It proposed to reuse the Boogie methodology [55] for class invariants in Object Oriented programs, based on an *ownership* relation between objects. The challenge was to combine this approach with the component-as-array memory modeling of *Jessie*. A solution is described in the master report [46] and a paper is in preparation.

6.2.8. 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. The case study currently used is a very low level specification of a secured boot loader called OSLO (Open Secure LOader), which includes some assembler routines. The objective is to specify finely this case study and identify the appropriate high-level constructions for specifying and verifying global security properties on a program.

6.2.9. 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.

C. Marché and N. Rousset implemented in *Krakatoa* the new constructions required for specifying algebraic models. This appears to be crucial to go further in the development of certified libraries. A roadmap for required future developments needed to fully support abstraction and refinement at the level of Java source code has been published [20]. Another article is under preparation on refinement techniques for Java programs.

6.2.10. Verification of MIX Programs

J.-C. Filliâtre proposed a methodology to verify programs written in assembly languages, and applied it to the verification of `MIX` programs. `MIX` is the ideal machine introduced by D. Knuth in *The Art of Computer Programming*. `MIX` programs are first annotated with logical annotations and then turned into purely sequential programs (with no more tests or jumps) on which classical techniques can be applied. Contrary to other approaches of verification of unstructured programs, the location of annotations is not imposed, but only the existence of at least one invariant on each cycle in the control flow graph. A prototype has been implemented and used to verify several programs from *The Art of Computer Programming*. This work has been presented at the *Journées en l'honneur de Donald E. Knuth* (Bordeaux, France) [34].

6.2.11. Guiding the Correction of Parameterized Specifications

Synthesizing inductive invariants is a key issue in many domains such as program verification or model based testing. However, few approaches help the designer in the task of writing a correct and meaningful model, where correction is used for consistency of the formal specification w.r.t. its inner invariant properties. In a joint work with F. Dadeau, J.-F. Couchot proposed [31] to ease the task of writing a correct and meaningful formal specification by using a panel of provers, presenting possibly unbounded data values to a finite one, (i.e. instantiating) without loss of generality, employing a set-theoretical constraint solver to conclude on the formula and, possibly, to produce a counter-example, and putting into practice some model-checkers to find a finite execution path, from the initial states to the invalid states corresponding to this instantiation. Thanks to this integration of techniques, the user is able to improve the quality of formal specifications.

6.3. Automated Deduction

Participants: Sylvain Conchon, Évelyne Contejean, Jean-François Couchot, Thierry Hubert, Johannes Kanig, Stéphane Lescuyer, Claude Marché.

6.3.1. Termination

C. Marché organized the fourth "Termination Competition", in collaboration with Hans Zantema of University of Eindhoven (http://www.lri.fr/~marche/termination-competition/). A report on the results of this competition has been published in the proceedings of the Workshop on Termination 2007 [40]. Moreover, a survey of all the past editions of the competition (also organized by C. Marché) has been presented at the RTA'07 conference [39].

C. Marché also continued to study termination for other computing formalisms than rewriting, extending previous results on Prolog programs to the membership equational programs paradigm used in the *Maude* system. This is done in collaboration with José Meseguer (University of Illinois at Urbana-Champaign, USA), Salvador Lucas (University of Valencia, Spain), Francisco Duran (University of Málaga, Spain) and Xavier Urbain (IEE, CNAM, France). This year, a new extension of the previous techniques has been proposed, to deal with rewriting modulo theories, and to support the so-called *operational termination*. This is published in the HOSC journal [18], and has been implemented into a new tool called MTT (http://www.lcc.uma.es/~duran/MTT/).

6.3.2. Decision Procedures

6.3.2.1. Polymorphic Logics

J.-F. Couchot extended the work of S. Lescuyer about the encoding of polymorphism: he showed how to encode PSL into multi-sorted logics such as the SMT-lib standard (thus allowing to use SMT provers such as Yices, Z3, CVC3 or RV-sat). Together, J.-F. Couchot and S. Lescuyer designed efficient reductions of polymorphism into both unsorted and many-sorted first order logic [33]. The efficiency keynote is to disturb the prover as little as possible, especially the internal decision procedures used for special sorts, e.g. integer linear arithmetic, to which a special treatment is applied. These reductions have been implemented in the *Why* platform [35] and successfully applied to many case studies.

6.3.2.2. The Ergo theorem prover

S. Conchon and É. Contejean fully formalized the core decision procedure $CC(X)$ of Ergo by inference rules. Together with J. Kanig, they gave some conditions to ensure soundness of the procedure [24].

6.3.2.3. Data structures

The implementation of the $CC(X)$ module requires an efficient union-find data structure. Due to the backtracking mechanism performed by the toplevel SAT-solver of Ergo, it is necessary to be able to come back to previous values of the union-find data structure. The usual and optimal union-find algorithm (due to Tarjan), is based on imperative data structures and thus is not amenable to backtracking. Sylvain Conchon and Jean-Christophe Filliâtre proposed an implementation of a persistent union-find data structure which appears to be almost as efficient as its imperative counterpart. Although it is persistent, this solution makes heavy use of imperative features. A formal proof of correctness has been done in the Coq proof assistant, especially to show the persistent nature of this solution. This work has been presented at JFLA 2007 [27] and at the SIGPLAN Workshop on ML 2007 [26].

S. Conchon and J.-C. Filliâtre generalized an idea present in the work described above 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 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 will be presented at ESOP'08 [28].

6.3.2.4. Dealing with Large Verification Conditions

Separation analysis is not always efficient enough to make VCs accessible to automatic theorem provers. On industrial C programs, such as case studies provided by Dassault Aviation, it remains some VCs that cannot be discharged by any SMT prover, yet valid. Typically, a VC contains a very large context of hypothesis and axioms are infinitely instantiated by useless terms from this context. J.-F. Couchot and T. Hubert designed a strategy to select relevant hypotheses in each VC [32]. The relevance of an hypothesis is the combination of syntactical and semantically separated dependency analysis implemented by some graph traversals.

The approach has been implemented in the *Why* platform and successfully applied on case studies from Dassault Aviation. As said in Section 6.2.4, it allowed to automatically solve 100% of the VCs generated for a first case study. It has then been experimented on a even larger case study (70,000 lines of code) and allowed to increase the ratio of VCs solved from 96.1% to 97.4%.

6.3.3. Automated proofs and certificates

6.3.3.1. Coccinelle and CiME's traces

É. Contejean has modelled permutations inside the Coccinelle library [17]. Permutations naturally arise when formally modeling rewriting in Coq, for instance RPO with multiset status and equality modulo AC. Moreover the needed permutations are up to an equivalence relation, and may be used to inductively define the same relation (equivalence modulo RPO). This is why the notion of permutation w. r. t. an arbitrary relation is

introduced. The advantages of this approach are a very simple inductive definition (with only 2 constructors), the adequacy with the mathematical definition, the ability to define a relation using recursively permutations up to this very relation, and a fine grained modularity (if `R` enjoys a property, so does `permut R`).

É. Contejean has also modelled and formally proved in Coccinelle some powerful criteria of termination: dependency pairs [54], the main modularity theorem for termination presented in the thesis of Urbain [85] as well as innermost termination, dependency pairs for it and its equivalence with standard termination in some specific cases [71].

É. Contejean and the CNAM participants of the A3PAT project, Pierre Courtieu, Julien Forest, Olivier Pons and Xavier Urbain have developed in CiME the production of termination traces, and in Coccinelle the modelling of rewriting relations and the "interpretation" of termination traces as termination certificates [47], [30]. This interpretation is based on dependency pairs criteria and uses either polynomial orderings or RPO. The well-foundedness of RPO and the fact that RPO is an ordering compatible with the rewriting relation has been fully proven in Coccinelle. Currently, they are able to certify the termination of 300 systems out of the 900 problems in the TPDB (termination problems data base). In 2007, CiME\Coccinelle was an entrant of the termination competition in the new category for certified termination proofs, and ranked the second place among the three participants.

6.3.3.2. Ergo's traces

S. Conchon, É. Contejean and J. Kanig have developed a mechanism to automate equality proofs generated by Ergo in Coq [48], [25]. The equality they considered is the combination of the pure theory of equality over uninterpreted symbols and a built-in decidable theory X. Their approach is intermediate between traditional complete proof reconstruction and verified provers. Reasoning about the theory X is implemented independently in Coq and in Ergo. This double blind computation enables both tools to interact only by lightweight traces about pure equalities which are produced by Ergo and interpreted by Coq. The Coq development follows the same modular architecture as Ergo. Currently, they have instantiated proof traces for the combination of the pure theory of equality and the theory of linear arithmetic.

6.4. Synchronous Programming

Participants: Alexandre Bertails, Dariusz Biernacki, Louis Mandel, Florence Plateau, Marc Pouzet.

6.4.1. Language extensions and type-systems

An important research activity of the field concerns the unification of data-flow and hierarchical automata, the two main programming models for designing real-time systems.

During year 2007, M. Pouzet has continued the work started in [2] [63] on the extension of synchronous languages, in collaboration with Jean-Louis Colaço (Esterel-Technologies). He also developed a new model of clock types to account for periodic real-time sampling.

6.4.2. Tools

During 2007, the development of the new version (V3) of the Lucid Synchrone compiler have been pursued. It is freely available at http://www.lri.fr/~pouzet/lucid-synchrone. A presentation of the language is given in a book chapter of a collection dedicated to real-time systems [15]. The language had served in 2007 for the development of a new programming language inside the Catia tool at Dassault-Systèmes and used for the simulation of industrial plants.

L. Mandel improved the development of ReactiveML, an extension of Objective Caml with reactive constructs [79]. During the period, several improvements of the compiler have been performed. A reimplementation is in progress, with an improved type system that guarantees the reactivity property. 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 [37].

6.4.3. Alternative synchronous models

During year 2007, Florence Plateau have proposed a largely simpler version of the original algorithm and algebraic properties of the N-synchronous model as introduced in colloboration with Albert Cohen, Christine Eisenbeis (INRIA Alchemy) and Marc Duranton (NXP Labs, Eindhoven). A journal version of [62] is under way.

Florence Plateau proposed a new clock calculus algorithm, using an abstraction technique. The idea is to forget the exact presence and absence instants of the stream by bounding the real clock by two very regular ones, much simpler to manipulate. This algorithm accepts only correct programs. It is not complete with respect to the original theory of N-synchrony but reduces the clock verification problem to a simpler problem, mainly the resolution of a set of linear constraints. Moreover, it can be adapted to type programs with non periodic clock provided they are bounded by periodic ones (which model jiitering in real-time systems).

6.4.4. Compiler certification

A new research activity, concerning the certification of a synchronous programs compiler inside the Coq proof assistant, has been started in September 2006.

In 2007, D. Biernacki and M. Pouzet have implemented MiniLucid, a prototype (yet efficient) compiler for Lustre in Ocaml. The compiler is based on program transformations and produces code in a simple object-based language that is then translated into a subset of C, Java or Ocaml. Each stage in the compiler is rigorously defined in the mathematical sense and therefore lends itself to formalization in Coq. A purely functional version of this compiler has been written in Coq. This work has been presented in [22].

It is clear that the certification of some stages of the compiler is easier to obtain by program extraction from Coq proofs of some logical properties (e.g., decidability of type inference), whereas of some others (e.g., translation into intermediate code) can be easier obtained by simply proving its correctness in Coq. Of the former category, D. Biernacki and A. Bertails have formalized in Coq the type system and the clock calculus, i.e., the bulk of the static semantics of Lustre, and through the Coq program extraction mechanism, we have obtained executable, certified Ocaml procedures that perform type and clock inference, a considerable part of the front-end of the compiler. Of the latter category, we have been working on reformulation of the standard relational semantics of Lustre into an operational formalism (in the style of Kahn's natural semantics) that seems to be more suitable for proving correctness of the program transformations used in the compiler (work in progress).

C. Paulin developed a Coq library for modeling Kahn's network [52] (deterministic parallel programs seen as stream transformers). It is based on a general theory of CPOs. The originality of this theory is to introduce computable least-upper-bounds using co-inductive definitions. The notion of Kahn's networks is defined as a cpo with good properties of closure by composition and recursion. This library could possibly be used for expressing the denotational semantics of synchronous programs.

7. Contracts and Grants with Industry

7.1. System@tic: PFC

Participants: Jean-Christophe Filliâtre, Thierry Hubert, Claude Marché, Christine Paulin, 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, Dassault Aviation 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. System@tic: GENCOD

Participant: Marc Pouzet.

This project is funded by FCE for two years, starting September 2007. The coordinator is Thales.

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

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

7.3. CIFRE grants

Participants: Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin, 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.

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.

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 http://www.rntl.org/projet/resume2005/cat.htm. 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: Malgorzata Biernacka, Sylvain Conchon, Évelyne Contejean, Johannes Kanig, 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://www3.iie.cnam.fr/~urbain/a3pat/

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 Ergo theorem prover inside the Coq proof assistant using an ad-hoc trace mechanism.

8.1.3. CerPAN

Participants: Sylvie Boldo, Jean-Christophe Filliâtre.

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.

8.1.4. ALIDECS

Participants: Marc Pouzet, Florence Plateau.

This project is funded by "ACI Sécurité et Informatique". The coordinator is M. Pouzet. http://www-verimag.imag.fr/~synchron/alidecs/.

Partners: INRIA PopArt, Grenoble; VERIMAG, Grenoble; INRIA MIMOSA, Sophia-Antipolis, CMOS University of Evry.

The objective of this project is to propose an integrated environment for the development of embedded systems. It is devoted to typical embedded systems expressed as closed loops. It is based on a set of languages for programming both the control part of the system as well as its environment.

8.1.5. SCALP

Participants: Christine Paulin, Yann Régis-Gianas.

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

It will start on january 2008 for 4 years; the coordinator is Yassine Lakhnech from VERIMAG, Grenoble.

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.

8.1.6. SIESTA

Participant: Marc Pouzet.

This project is funded by ANR (program RNTL), on automation of the test of embedded systems designed in SCADE and SIMULINK. It starts in January 2008, and is coordinated by I. Parissis from LIG, Grenoble.

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 and will last until april 2008. It is a continuation of a number of successful European projects (ESPRIT BRA 6453, 1992 - 1995, ESPRIT working group 21900, 1997 - 1999 and IST working group 29001, 2000 - 2003) http://www.cs.chalmers.se/Cs/Research/Logic/Types/

The project involves 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 is 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. PAI Polonium Types, Proofs and Correct Programs

We participate to the PAI "Types, Proofs and Correct Program" which is a joint program (2006-2007) coordinated by J.-P. Jouannaud (LIX, Ecole Polytechnique) and P. Urzyczyn (University of Warsaw, Poland).

8.3. International initiatives

8.3.1. Taiwan

We have a collaboration with National Taiwan University and Academia Sinica in Taipei (Taiwan). In May, we submitted an ORCHID project on Formal Methods for Software Security.

8.4. Visits, researcher invitation

8.4.1. Visits

Marc Pouzet was invited at the University of Bamberg (Germany) for a week by Prof. Mendler in July 2006 and November 2007.

He also spend two months during spring 2007 at NXP Labs. in Eindhoven (The Netherlands), under the European Marie-Curie contract PSYCHES in collaboration with INRIA team Alchemy and NXP.

8.4.2. Invitations

Dr Jacek Chrząszcz and Dr Daria Walukiewicz from Warsaw University (Poland) visited the Proval team for one month, in June 2007.

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

C. Paulin is vice-president of the project committee of INRIA Futurs and member of the national evaluation board of INRIA. In 2007, she participated to the jury of CR2 hiring committee at INRIA Futurs-Bordeaux and INRIA Futurs-Lille as well as the national DR2 hiring committee.

C. Marché was member of the jury of CR1 hiring committee in 2007 at INRIA Futurs.

9.1.3. Collective responsibilities outside INRIA

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

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

C. Paulin is an elected member of the board (*"conseil d'administration"*) of University Paris-Sud 11.

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 2007.

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/.

9.1.4. Event organization

C. Marché organized the 4th "Termination Competition" in June 2007 http://www.lri.fr/~marche/termination-competition/

S. Boldo organized a session of a workshop of the working group "Arithmétique" of the GDR "Informatique Mathématique" in January 2007.

É. Contejean organized the 21th International Workshop on Unification in June 2007 http://hal.inria.fr/UNIF07/.

Together with Th. Coquand, G. Dowek, F. Fages, X. Leroy and D. Rémy, C. Paulin organised in June 2007 the colloquium in honor of Gérard Huet 60th birthday http://www.lix.polytechnique.fr/~dowek/GH/.

9.1.5. Learned societies

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

9.1.6. Editorial boards

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 is a member of the program committee of the 7th International Workshop on Rewriting Logic and its Applications (WRLA'08).

C. Paulin was a member of the program committee of the 14th Workshop on Logic, Language, Information and Computation (WoLLIC'2007, Rio de Janeiro, Brazil) and of the 20th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2007, Kaiserslautern, Germany). She is participating to the program committee of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008, Montreal, Quebec, Canada). She is also Program Chair and will organize the 9th International Conference on Mathematics of Program Construction (MPC 2008).

S. Conchon is a member of the program committee of 3ème Journée Francophone sur le Développement de Logiciels Par Aspects (JFDLPA 2007).

J.-C. Filliâtre was a member of the program committee of the 20th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2007, Kaiserslautern, Germany) and of the 2nd Automated Formal Methods workshop (AFM07, Atlanta, USA). He will be a member of the program committee of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008, Montreal, Quebec, Canada).

É. Contejean is the program committee chair of the International Workshop on Unification (UNIF 2007, June 29, Paris, France).

Marc Pouzet is the co-chair of the workshop on synchronous languages (SLAP) in 2006. He was member of the program committe of the "Journées Francophones des Langages Applicatifs" (JFLA) 2008, of the conference "Modélisation des systèmes réactifs" MSR 2006 and 2007, FMGALS 2007, Real-Time and Network Systems Conference 2007. He is member of the steering committee of the workshop on Model-driven Higher-level Programming of Embedded Systems (SLA++P) since 2006. He will be member of the program committee of RTNS 2008.

9.1.8. Invited Presentations

J.-C. Filliâtre was invited speaker at CORTA 2007 (Covilhã, Portugal).

C. Marché was invited speaker at PLPV 2007 (Freiburg, Germany).

9.1.9. Participation to PhD juries

C. Marché reviewed the PhD manuscript of Mariela Pavlova (University of Nice, January 19th, 2007) and the manuscript of Yann-Régis-Gianas (University Paris 7, November 29th, 2007). He was member of the PhD jury of Nicolas Stouls (Institut National Polytechnique de Grenoble, December 14th, 2007).

C. Paulin was member of the PhD jury of Hichem Houissa (University Paris-Sud 11, May 22nd, 2007), Kostantinos Chatzikokolakis (Ecole Polytechnique, Oct 26th, 2007), Pierre Sennelart (University Paris-Sud 11, Dec 12th, 2007). She reviewed the manuscript and was member of the PhD jury of Julien Groslambert (University of Franche-Comté, Sept. 14th, 2007),

M. Pouzet was member of the Phd jury of Sébastien Labbe (University of Paris-6, September 25th 2007). He reviewed the PhD manuscript of Frédéric Dabrowski (University of Paris 7, June 22nd 2007) and of Jean-Baptiste Note (Université of Paris 6 and Ecole Normale Supérieure, October 31st 2007).

9.2. Teaching

9.2.1. Supervision of PhDs and internships

S. Boldo supervised the internship of A. Viel (ÉNS student) (proofs of floating-point summation programs, April to September 2007).

S. Conchon and É. Contejean supervised the internship of Johannes Kanig (2nd year of the *Master Univ. Dresden, Germany*, sep 2006-aug 2007). S. Conchon and É. Contejean supervise the Ph.D. thesis of Stéphane Lescuyer (started September 2007) (complete certification of an automated theorem prover dedicated to program verification).

J.-C. Filliâtre supervised the internship of B. Vadon (design and implementation of a graph editor based on hyperbolic geometry, February to June 2007). J.-C. Filliâtre is supervising the Ph.D. thesis of Johannes Kanig (started September 2007).

C. Marché supervises PhD of Thierry Hubert (CIFRE co-supervised by E .Ledinot, Dassault aviation), Nicolas Rousset (CIFRE co-supervised by B. Chetali, Gemalto), Yannick Moy (CIFRE co-supervised by P. Crégut, France Télécom R&D), and Romain Bardou (ÉNS Cachan, since September 2007). He supervised the master thesis of Romain Bardou (Feb. to Aug. 2007).

C. Paulin supervises the Ph. D. thesis of Matthieu Sozeau (programming with dependent types in Coq) and Wendi Urribarrí (towards certified libraries). Together with Franck Védrine (CEA-LIST), she supervises the PhD of Nicolas Ayache (Verification of System C programs) at CEA LIST.

M. Pouzet supervises the PhD of Florence Plateau (LRI, since sept. 2005). He is the co-advisor (with Alain Girault) of the PhD. thesis of Gwenael Delaval (INRIA PopArt Grenoble, start in sept. 2004).

9.2.2. Graduate courses

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

  In 2006–2007, J.-C. Filliâtre lectured in the course on "Constructive Proofs" (12h).

  In 2007-2008, É. Contejean lectured on advanced rewriting (12h) in the course on "Automated Deduction".

  In 2007-2008, C. Paulin lectured (12h) in the course on "Proof assistants".

  In 2007-2008, M. Pouzet lectured (15h) in the course on "Synchronous System".
- 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).
- C. Paulin and J.-C. Filliâtre lectured at the Types Summer School (August 19-31th, 2007, Bertinoro, Italy), respectively on the Coq proof assistant and the *Why* platform (9h course + 5h practical lab).

9.2.3. Other Courses

S. Conchon, L. Mandel 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 (64h per year) as *"monitrice"* 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).

9.3. Industrial Dissemination

C. Marché presented the activities of the team, and gave a demo of the *Why* platform, at the "Journée INRIA-industrie Confiance et Sécurité" in October 2007 http://www.inria.fr/valorisation/rencontres/securite/programme.fr.html.

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 (October 12-13, 2007). F. Plateau, Y. Moy, S. Boldo, J. Kanig and N. Stouls animated this activity. This involved the programmation of a computer game with a graphical interface and an optimal computer resolution. One hundred visitors (students, families) attended their activity.

10. Bibliography

Major publications by the team in recent years

[1] 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 Artificial Intelligence, vol. 4130, Springer-Verlag, August 2006.

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

[4] J.-C. FILLIÂTRE, P. LETOUZEY. *Functors for Proofs and Programs*, in "Proceedings of The European Symposium on Programming, Barcelona, Spain", Lecture Notes in Computer Science, vol. 2986, April 2004, p. 370–384, http://www.lri.fr/~filliatr/ftp/publis/fpp.ps.gz.

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

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

[7] 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", IEEE Comp. Soc. Press, September 2005.

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

[9] 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.

[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° 1–2, 2004, p. 89–106, http://krakatoa.lri.fr.

Year Publications

Articles in refereed journals and book chapters

[11] P. AUDEBAUD, C. PAULIN-MOHRING. *Proofs of Randomized Algorithms in Coq*, in "Science of Computer Programming", to appear, 2007.

[12] D. BIERNACKI, O. DANVY, K. MILLIKIN. *A Dynamic Continuation-Passing Style for Dynamic Delimited Continuations*, in "Transactions on Programming Languages and Systems", to appear, 2007.

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

[14] S. BOLDO, G. MELQUIOND. *Emulation of FMA and correctly-rounded sums: proved algorithms using rounding to odd*, in "IEEE Transactions on Computers", to appear, 2007, http://hal.inria.fr/inria-00080427.

[15] P. CASPI, G. HAMON, M. POUZET. *Synchronous Functional Programming with Lucid Synchrone*, in "Real-Time Systems: Models and verification — Theory and tools", ISTE, 2007.

[16] S. CONCHON, J.-C. FILLIÂTRE, J. SIGNOLES. *Designing a Generic Graph Library using ML Functors*, in "Trends in Functional Programming", to appear, vol. 8, Intellect, 2007.

[17] É. CONTEJEAN. *Modelling permutations in Coq for Coccinelle*, in "Rewriting, Computation and Proof", H. COMON-LUNDH, C. KIRCHNER, H. KIRCHNER (editors), Lecture Notes in Computer Science, Jouannaud Festschrift, vol. 4600, Springer, 2007, p. 259–269.

[18] F. DURÁN, S. LUCAS, J. MESEGUER, C. MARCHÉ, X. URBAIN. *Proving Operational Termination of Membership Equational Programs*, in "Higher-Order and Symbolic Computation", to appear, 2007.

[19] L. MANDEL, M. POUZET. *ReactiveML : un langage fonctionnel pour la programmation réactive*, in "Technique et Science Informatiques (TSI)", to appear, 2007.

[20] C. MARCHÉ. *Towards Modular Algebraic Specifications for Pointer Programs: a Case Study*, in "Rewriting, Computation and Proof", H. COMON-LUNDH, C. KIRCHNER, H. KIRCHNER (editors), Lecture Notes in Computer Science, vol. 4600, Springer, 2007, p. 235–258.

Publications in Conferences and Workshops

[21] M. BIERNACKA, D. BIERNACKI. *Formalizing Constructions of Abstract Machines for Functional Languages in Coq*, in "7th International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2007), Paris, France", June 2007.

[22] D. BIERNACKI, J.-L. COLAÇO, M. POUZET. *Modular Compilation from Synchronous Block-diagrams*, in "Workshop on Automatic Program Generation for Embedded Systems (APGES), Salzburg, Austria", Embedded System Week, october 2007.

[23] 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.

[24] S. CONCHON, É. CONTEJEAN, J. KANIG. *CC(X): Efficiently Combining Equality and Solvable Theories without Canonizers*, in "SMT 2007: 5th International Workshop on Satisfiability Modulo", S. KRSTIĆ, A. OLIVERAS (editors), 2007.

[25] S. CONCHON, É. CONTEJEAN, J. KANIG, S. LESCUYER. *Lightweight Integration of the Ergo Theorem Prover inside a Proof Assistant*, in "AFM07 (Automated Formal Methods)", J. RUSHBY, N. SHANKAR (editors), 2007.

[26] 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.

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

[28] S. CONCHON, J.-C. FILLIÂTRE. *Semi-Persistent Data Structures*, in "17th European Symposium on Programming (ESOP'08), Budapest, Hungary", To appear, April 2008.

[29] 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, http://www.lri.fr/~filliatr/ftp/publis/ocamlgraph-tfp07.ps.

[30] 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", F. WOLTER (editor), Lecture Notes in Artificial Intelligence, Springer, September 2007.

[31] J.-F. COUCHOT, F. DADEAU. *Guiding the Correction of Parameterized Specifications*, in "Integrated Formal Methods, Oxford, UK", Lecture Notes in Computer Science, vol. 4591, Springer, July 2007, p. 176–194.

[32] 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, http://lifc.univ-fcomte.fr/~couchot/IMG/pdf_couchot_hubert.pdf.

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

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

[35] 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, n^{o} 4590, Springer, July 2007.

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

[37] L. MANDEL, L. MARANGET. *Programming in JoCaml*, in "17th European Symposium on Programming (ESOP'08), Budapest, Hungary", To appear, April 2008.

[38] 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, New York, NY, USA", ACM, 2007, p. 1–2.

[39] C. MARCHÉ, H. ZANTEMA. *The Termination Competition*, in "18th International Conference on Rewriting Techniques and Applications (RTA'07), Paris, France", F. BAADER (editor), Lecture Notes in Computer Science, Springer, June 2007.

[40] C. MARCHÉ, H. ZANTEMA, J. WALDMANN. *The Termination Competition 2007*, in "Extended Abstracts of the 9th International Workshop on Termination, WST'07", D. HOFBAUER, A. SEREBRENIK (editors), June 2007, http://www.lri.fr/~marche/termination-competition/.

[41] Y. MOY, C. MARCHÉ. *Inferring Local (Non-)Aliasing and Strings for Memory Safety*, in "Heap Analysis and Verification (HAV'07), Braga, Portugal", March 2007.

[42] Y. MOY. *Union and Cast in Deductive Verification*, in "Proceedings of the C/C++ Verification Workshop", vol. Technical Report ICIS-R07015, Radboud University Nijmegen, July 2007.

[43] 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, to appear, Springer, January 2008, http://www.lri.fr/~moy/publis/moy08vmcai.pdf.

[44] M. SOZEAU. *Program-ing Finger Trees in Coq*, in "ICFP'07: Proceedings of the 2007 ACM SIGPLAN International Conference on Functional Programming", ACM Press, 2007, p. 13–24, http://portal.acm.org/citation.cfm?doid=1291151.1291156.

[45] M. SOZEAU. *Subset coercions in Coq*, in "Types for Proofs and Programs, International Workshop, TYPES 2006, Revised Selected Papers", T. ALTENKIRCH, C. MCBRIDE (editors), Lecture Notes in Computer Science, vol. 4502, Springer, 2007, p. 237–252, http://www.springerlink.com/content/8p35448426n2n41k/.

Internal Reports

[46] R. BARDOU. *Invariants de classe et systèmes d'ownership*, Master's thesis, Master Parisien de Recherche en Informatique, 2007, http://romain.bardou.fr/papers/stage2007r.pdf.

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

[48] J. KANIG. *Certifying a Congruence Closure algorithm in Coq using Traces*, Diplomarbeit, Technische Universität Dresden, April 2007.

[49] Y. MOY. *Checking C Pointer Programs for Memory Safety*, Research Report, n^{o} 6334, INRIA, October 2007, http://hal.inria.fr/inria-00181950.

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

Miscellaneous

[51] S. BOLDO. *Kahan's algorithm for a correct discriminant computation at last formally proven*, 2007, http://hal.inria.fr/inria-00171497/.

[52] C. PAULIN-MOHRING. *A constructive denotational semantics for Kahn networks in Coq*, To be published in the volume in memory of Gilles Kahn, 2007, http://www.lri.fr/~paulin/PUBLIS/paulin07kahn.pdf.

References in notes

[53] 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-Verlag, July 2005.

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

[55] 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, 2004, p. 27-56.

[56] A. BENVENISTE, P. CASPI, S. 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.

[57] 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.

[58] 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.

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

[60] M. CLAVEL, F. DURÁN, S. EKER, P. LINCOLN, N. MARTÍ-OLIET, J. MESEGUER, J. QUESADA. *Maude: specification and programming in rewriting logic*, in "Theoretical Computer Science", vol. 285, n^{o} 2, August 2002, p. 187–243.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

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

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

[68] 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.

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

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

[71] 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-Verlag, July 1996, p. 93–107.

[72] 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.

[73] B. JACOBS. *Coalgebraic Reasoning about Classes in Object-Oriented Languages*, in "Workshop Coalgebraic Methods in Computer Science (CMCS)", Electronic Notes in Computer Science, vol. 11, 1998.

[74] 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-Verlag, July 2004.

[75] 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.

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

[77] P. LETOUZEY. *A New Extraction for Coq*, in "TYPES 2002", H. GEUVERS, F. WIEDIJK (editors), Lecture Notes in Computer Science, vol. 2646, Springer, 2003, http://www.pps.jussieu.fr/~letouzey/research.fr.html.

[78] P. LETOUZEY. *Programmation fonctionnelle certifiée: l'extraction de programmes dans l'assistant Coq*, Thèse de Doctorat, Université Paris-Sud, July 2004, http://www.pps.jussieu.fr/~letouzey/research.fr.html.

[79] 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.

[80] C. MARCHÉ. *Preuves mécanisées de propriétés de programmes*, Mémoire d'habilitation, Université Paris-Sud, December 2005.

[81] 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", September 2006, http://www.lri.fr/~marche/termination-competition/.

[82] 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-Verlag, July 2000, p. 270–273, http://bibiserv.techfak.uni-bielefeld.de/talp/.

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

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

[85] 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.

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

[87] B. WERNER. *On the strength of proof-irrelevant type theories*, in "3rd International Joint Conference on Automated Reasoning", 2006.