INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Proval

Proof of programs

Futurs

THEME SYM

Activity Report

2006

Table of contents

1. Team

The Proval project is a PCRI project common to INRIA-Futurs, CNRS and Université Paris-Sud 11. It involves researchers from LRI (Laboratoire de Recherche en Informatique, UMR 8623).

Team Leader
Christine Paulin [Professor University Paris-Sud 11, secondment INRIA since Sep. 06, HdR]

Team Vice-Leader
Claude Marché [Assistant Professor University Paris-Sud 11, DR INRIA since Sep. 06, HdR]

Administrative assistant
Marie-Carol Lopez [until Oct. 06, TR INRIA]
Stéphanie Meunier [since Nov. 06, TR INRIA]

Technical assistant
Aurélien Oudot [Temporary Engineer INRIA, since Sep. 06]

Research Scientists
Marc Pouzet [Professor University Paris-Sud 11, HdR]
Sylvie Boldo [CR INRIA]
Évelyne Contejean [CR CNRS]
Jean-Christophe Filliâtre [CR CNRS]
Sylvain Conchon [Assistant Professor Université Paris-Sud 11]

Ph. D. students
June Andronick [CIFRE, Axalto, Louveciennes, until Mar. 06]
Alexandre Bertails [MENRT, University Paris-Sud 11, since Oct. 06]
Thierry Hubert [CIFRE, Dassault aviation, Suresnes]
Yannick Moy [CIFRE, France Télécom, Lannion]
Nicolas Oury [MENRT, University Paris-Sud 11, INRIA Contract Sep-Oct. 06]
Florence Plateau [MENRT, University Paris-Sud 11]
Nicolas Rousset [CIFRE, Gemalto, Louveciennes]
Julien Signoles [MENRT, University Paris-Sud 11, until Aug. 06]
Matthieu Sozeau [MENRT, University Paris-Sud 11]
Wendi Urribarrí [MENRT, University Paris-Sud 11, since Nov. 06]

Post-doctoral fellows
Malgorzata Biernacka [Post-doc CNRS, since Sep. 06]
Dariusz Biernacki [Post-doc INRIA, since Sep. 06]
Jean-François Couchot [Post-doc INRIA, since Sep. 06]

Student interns
Christophe Avenel [1st year Master Informatique Univ. Paris-Sud 11, Jan. to May 06]
Alexandre Bertails [2nd year Master Parisien de Recherche en Informatique, Mar. 06 to Sep. 06]
Medhi Dogguy [1st year Master Informatique Univ. Paris-Sud 11, Jan. to May 06]
Johannes Kanig [2nd year Master Univ. Dresden, Germany, Sep. 06 to Feb. 07]
Stéphane Lescuyer [2nd year Master Parisien de Recherche en Informatique, Mar. 06 to Sep. 06]

2. Overall Objectives

2.1. Overall Objectives

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 project, formed after the LogiCal project, 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 (pointers, random computations, floating point arithmetic, semantics ...).

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

When sophisticated tools are used for analysing 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 decision procedures, compilers or verification condition generators 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 our own tools.

The project develops a generic environment (Why) for proving programs; Why generates sufficient conditions for the 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 below, following the main 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 the four topics.

3. Scientific Foundations

3.1. Higher-Order Functional Languages

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

Participants: Sylvie Boldo, Jean-Christophe Filliâtre, Nicolas Oury, Christine Paulin, Julien Signoles, Matthieu Sozeau.

Higher-order strongly typed programming languages such as Objective Caml help to improve 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. The 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. Program extraction

Regarding purely functional ML programs, the technique of extraction has been around for years in the Coq proof assistant. It consists in getting program by erasing logical parts in constructive proofs of specifications. Such programs are then correct by construction. During his PhD thesis, P. Letouzey designed and implemented a new extraction mechanism for the Coq system [62], [63], much more powerful than the old version. With this new extraction, J.-C. Filliâtre and P. Letouzey could verify ML 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 features for the Coq system, and has been used by several teams around the world in order to get efficient certified code [61].

3.1.2. Developing correct ML programs

We are also considering other ways to tackle ML programs. J. Signoles worked for his PhD on an extension of ML with refinement, a methodology usually applied to imperative programming languages. The key idea is to mix types and expressions into a single syntactical entity. It leads to under-determinism, as usual in methods based on refinement, but simultaneously to dependent types. The usual notion of typing becomes a particular case of a more general notion of refinement between two programs.

We are also pursuing the idea of using directly a powerful type theory as a programming language where dependent types are expressing the specification. With this approach, well-typed programs are correct with respect to their specification, but they also contain extra correctness informations which are required for type-checking. We are studying alternative input languages where the system infers automatically part of this information in a user-friendly manner. Our main contributions in this area are the study of completeness of patterns with dependent types and also programming with subset types.

3.1.3. Randomized algorithms

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. See Section 6.1.3.

3.1.4. Floating-point programs

Many industrial programs (weather forecasts, plane trajectories, simulations...) use floating-point computations, typically double precision floating-point numbers. 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 may go wild due to exceptional behaviors (like division by zero) [1]. 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. The method is to use the Why tool. The user or the tool will provide annotations in order to generate the proof obligations corresponding to “there is no overflow” or “the final error is less than...”. 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.

3.1.5. 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: Sylvie Boldo, Jean-Christophe Filliâtre, Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin.

A foundation step of the project is the PhD thesis of Jean-Christophe Filliâtre [5] which proposes to establish soundness of a program with imperative features (assignments, while loops, but also exceptions and exception handlers) by use 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 [17].

Following this thesis, a new tool called Why [55] 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*: 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 verification conditions. It was early remarked that this tool was independent of Coq, because the proof obligations 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 [57]. The approach is based on a translation from annotated Java programs into the specific language of Why, we then can reuse Why's verification conditions generation mechanism and choose between different provers for establishing these obligations.

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 [56].

3.2.1. The Why/Krakatoa/Caduceus platform

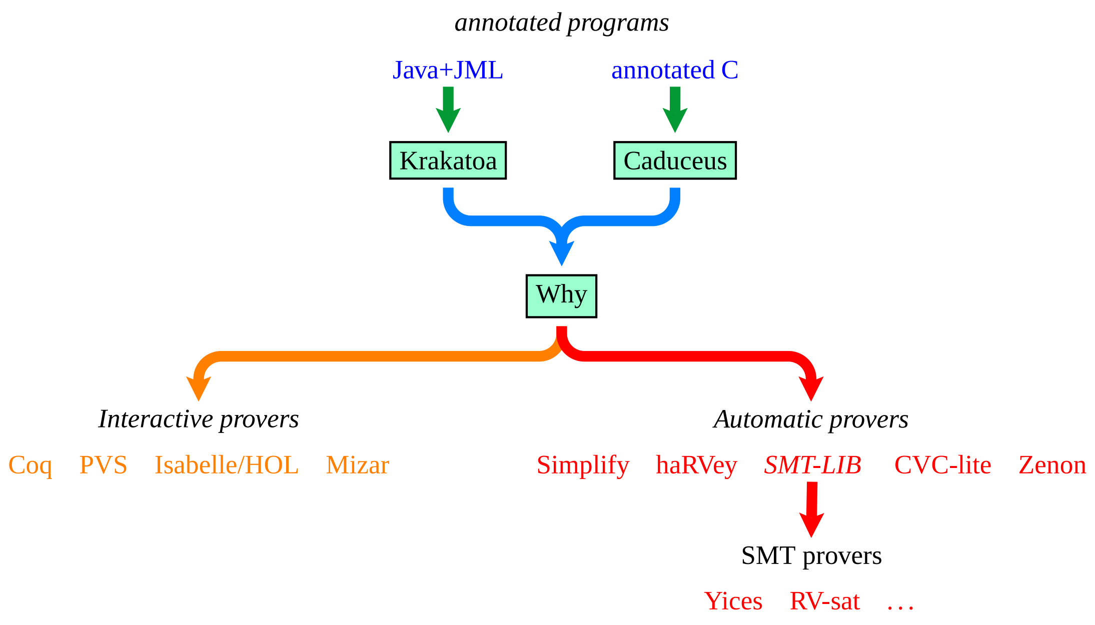

Figure 1. Overview of the platform architecture

We develop a platform combining several tools of our own, and external ones. The tool playing the central role in our platform is Why [55], 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 verification conditions, the validity of which ensures correctness of the program with respect to the original specification. In Why, these obligations are first-order formula which 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 CVC-lite, Simplify, Yices or haRVey. 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 do 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 [65]. 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. 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 [60] proposed by Gemalto company, this case study have been conducted in collaboration with LOOP and Jive groups. Banking applications are concerned with security problems which can be the confidentiality and protection of datas, authentication...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 [49] 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 [45],[30].

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. We are also collaborating with Gemalto on the proof of part of the smart card operating system, written in C. 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, Evelyne Contejean, 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 are under study in recent years are termination proofs techniques, the issue of combination of decision procedures, and generation of proof traces.

On termination topic, we have been interested in the design of new techniques which can be automated. A fundamental result of ours is a new criterion for checking termination *modularly* and *incrementally* [68], and further generalizations [9]. These new criteria and methods have been implemented into the CiME2 rewrite toolbox [3]. Around 2002, several new 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 new direction of research on termination techniques was also to apply our new approaches for rewriting to other computing formalisms, first to Prolog programs [67] and then to membership equational programs [54], a paradigm used in the *Maude* system [50].

Our research related to combination of decision procedures was initiated by a result [58] 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],[15], .

We are now focusing on the integration of decision procedures within user-assisted proof environments: in particular in the Coq system [36].

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.

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 [53]. We are currently experimenting with this idea in the domain of termination and of decision procedures. This is also the main purpose of the ANR project A3PAT: the next version of the CiME toolbox should provide certificates for equality proofs by rewriting, termination and local confluence of rewriting systems and so on.

3.4. Synchronous Programming

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

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

The goal is to propose high-level languages for the development of critical embedded systems with high temporal constraints [46], [59], [47], [48]. The research activities concern the extension of synchronous languages with abstraction mechanisms (e.g., modularity, functionality), the development of type systems or dedicated static analysis and the links between data-flow (*à la* SCADE or Simulink) and imperative models (*à la* Esterel or StateFlow). The results are implemented in the programming language Lucid Synchrone and the object of a long-term collaboration with the SCADE team of Esterel-Technologies. Marc Pouzet has been involved in the design of SCADE V6, the new evolution of SCADE.

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 which are developed inside the companies or by external partners.

Many smartcards in mobile phones are based on a (small) Java virtual machine. The card is supposed to execute applets which 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 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 behaves according to the requirements. We have a collaboration with Axalto 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: Evelyne Contejean [correspondant], 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 tool

Keywords: *Monadic functional interpretation*, *Verification conditions*.

Participants: Jean-Christophe Filliâtre [correspondent], Jean-François Couchot, Mehdi Dogguy, Claude Marché.

The Why tool produces verification conditions from annotated programs given as input. It differs from other systems in that it outputs conditions for several existing provers (including Coq but also PVS, HOL-light, Mizar, Simplify and haRVey). Why has been used by external researchers in published verifications of non-trivial algorithms (Efficient square root used in GMP, Knuth's algorithm for prime numbers). Why is also aimed at being used as a back-end for other tools dealing with real programming languages, like for Krakatoa and Caduceus presented below.

Distributed as open source since July 2002, under the GPL licence, at URL http://why.lri.fr/

5.3. The Krakatoa tool

Keywords: *Formal verification*, *Java Card*, *Java modeling language*, *Java programming language*.

Participants: Claude Marché [correspondent], Christine Paulin, Nicolas Rousset.

Krakatoa is a prototype verification tool for Java programs, using Why as a back-end. Distributed as open source since March 2003, at URL http://krakatoa.lri.fr. It is currently under experimentation at Gemalto company.

It has been used for teaching in 2004-2005 (third year, University of Evry, France)

5.4. The Caduceus tool

Keywords: *ANSI C programming language*, *Formal verification*.

Participants: Jean-Christophe Filliâtre [correspondent], Thierry Hubert, Claude Marché, Yannick Moy.

Caduceus is a prototype verification tool for C programs, using Why as a back-end. Distributed since 2004 as open source, under the GPL licence, at URL http://caduceus.lri.fr. It is currently under experimentation at Gemalto company, at Dassault Aviation company, and at CEA.

It is 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)

5.5. The Ergo theorem prover

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

Participants: Sylvain Conchon [correspondent], Évelyne Contejean, Johannes Kanig.

Ergo is an automated theorem prover, whose development started in 2006. It is distributed as open source, under the CeCILL-C licence, at URL http://ergo.lri.fr. It is available as a back-end prover in the Why platform.

5.6. Lucid Synchrone

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

Participant: Marc Pouzet [correspondent].

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 some features from ML languages.

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 compiler (the industrial version of Lustre) developped at Esterel-Technologies.

5.7. Bibtex2html

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

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

Bibtex2html is a generator of HTML pages of bibliographic references. Distributed as open source, under the GPL licence, at URL http://www.lri.fr/~filliatr/bibtex2html/. Bibtex2html is distributed as a package in most Linux distributions. A Google search of the appropriate banner string reveals that (on December 2006) around 30000 web pages have been generated using Bibtex2html.

6. New Results

6.1. Higher-order functional programs

6.1.1. Programming with dependent types

Developing programs with dependent types in the Coq system is not as easy as writing a functional program (for instance in Ocaml or Haskell).

There are many problems due to the weakness of convertibility, N. Oury [12] obtained a theoretical result linking extensional and intensional type theories which is a good basis for proposing extensions of CIC while keeping a small safe kernel. He also proposed [42] an original strategy for establishing the completeness of pattern-matching with dependent types using a method related to abstract interpretation which computes an approximation of the closed terms inhabiting a pattern.

M. Sozeau worked on a specific language for writing programs when dependent types are introduced via subset types in the spirit of PVS [32]. He designed a translation from simply typed terms into partial proofs of rich dependent types that he proved to be correct modulo extra convertibility rules in the logic. He also implemented a prototype of his method which is available in the current version of Coq. This permits to separate coding and proving phases while developing complex certified programs in the proof assistant.

6.1.2. ML with refinement

In his PhD thesis supervised by J.-C. Filliâtre, J. Signoles introduced an extension of the Ocaml language with non-deterministic constructions [35], [13]. A type is considered to be an ordinary program construction representing an arbitrary value of this type, which can be inserted in any sub-expression. The type language can integrate dependent types to make explicit the expected logical properties of values. The classical typing relation between a program and its type becomes a particular case of *refinement* between two programs.

6.1.3. Randomized programs

C. Paulin in collaboration with Ph. Audebaud from the Marelle team, proposed a method for modeling probabilistic programs in Coq. This work has been presented at the MPC 2006 conference [20]. They use a monadic interpretation of probabilistic programs as probability measures. They developed the corresponding Coq library, describing the interval $[0, 1]$, probability measures and an axiomatic semantics for programs [33]. This library has been used for the verification of small examples like implementations of Bernoulli or Binomial laws.

6.1.4. Floating-point programs

During and after her stay at the National Institute for Aerospace (NIA, Hampton, VA) between February and April 2005, S. Boldo has been working on floating-point arithmetic properties in PVS. This work is both a port of the former Coq formalisation and a new result about polynomial evaluation that is formally proved to be faithful on mild assumptions (met for example when computing elementary functions) [23].

S. Boldo has investigated the difficulties of a strong justification of the validity of old well-known algorithms for a generic radix and even when Underflow or Overflow occurs. This includes the Veltkamp algorithm, used to split a float into an upper part and a lower part and the Dekker algorithm, used to compute the exact error of a floating-point multiplication. This work has been presented at IJCAR 2006 [21].

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, M. Daumas and G. Melquiond therefore contributed to the formal proof of this algorithm [22].

6.1.5. Programming techniques

In the context of the development of a graph library for the Objective Caml programming language, Jean-Christophe Filliâtre studied the idiom of *backtracking iterators*. These are first-order data structures to iterate over the elements of a collection, similar to the iteration classes which can be found in object-oriented programming languages, but which are based on persistent data structures. Thus one can backtrack to a previous point of the iteration or conduct several iterations in parallel. Several backtracking iterators over binary trees are implemented and strong connections with G. Huet's *Zipper* are exhibited. This work has been presented at JFLA 2006 [29] and at the SIGPLAN Workshop on ML 2006 [28].

6.1.6. Tool development

The high-level language designed M. Sozeau for developing programs with subset types is part of the new Coq distribution [36]. C. Paulin also proposed and implemented a new criteria for parameters in inductive definitions introducing a notion of recursive parameters which leads to a more convenient primitive elimination scheme.

6.2. Proof of imperative and object-oriented programs

Participants: June Andronick, Sylvie Boldo, Jean-Christophe Filliâtre, Thierry Hubert, Claude Marché, Yannick Moy, Christine Paulin, Nicolas Rousset.

6.2.1. Graphical User Interfaces

During a graduate internship supervised by Jean-Christophe Filliâtre, Mehdi Dogguy improved the graphical user interface GWhy. This tool displays the verification conditions generated by the Why tool and their connections to the source files (i.e. the C source files when used on files generated by Caduceus). It also allows the user to call several decision procedures on each of these verification conditions, in parallel. Results are displayed in a graphical way, so that the user immediately sees whether a function has been proved correct or not. When not, the tool can be used to analyze the reasons of the proof failure, in order to modify the program and/or its specification.

A. Oudot started in October 2006 the development of a new user interface for the Why/Krakatoa/Caduceus platform, integrated into the *Eclipse* environment, a widely used framework for Java development tools. Such an interface should improve the visibility and the usability of our tools. These interface will be generic, in the sense that it will support both Krakatoa and Caduceus but also other tools developed in the CAT project.

6.2.2. Floating-Point C Programs

Sylvie Boldo and Jean-Christophe Filliâtre introduced a methodology to perform formal verification of floating-point C programs. It extends the Caduceus tool with new annotations specific to floating-point arithmetic. The Caduceus first-order logic model for C programs is extended accordingly. Then verification conditions expressing the correctness of the programs are obtained in the usual way and can be discharged interactively with the Coq proof assistant, using an existing Coq formalization of floating-point arithmetic by Daumas, Rideau and Théry. This methodology is already implemented and has been successfully applied to several short floating-point programs. This work has been submitted to ARITH 18 [39].

6.2.3. 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: shortly speaking, pointers as parameters of functions may have a *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 proof obligations solved automatically by the Simplify automatic theorem prover. The 10 remaining ones have been proved in the Coq proof assistant.

An article describing this approach will be submitted soon.

6.2.4. 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 devised a new modular, contextual and idiomatic approach to memory safety for C pointer programs. It uses a combination of abstract interpretation and automatic proof that makes it both automatic and user-oriented. An article describing this approach will be submitted soon. Y. Moy implemented this analysis in the Caduceus tool. This implementation is roughly 10,000 lines of Ocaml for the plugin part inside Caduceus, and a few hundreds lines of C to patch the publicly available *octagon abstract domain* library of A. Miné [66].

6.2.5. Java Card transactions

In her thesis [11], J. Andronick investigated using the Caduceus tool for proving C code implementing the system on a Java Card. She showed how to combine a local proof of correctness of code with the systematic generation of a high-level model for checking global properties of the system. She applied this technique for checking that low-level operations like memory-block erasing will resist a card tear [19]. She also investigated alternative models in Caduceus for handling union types and casts in order to have a sound representation of the C programs embedded on Javacards.

N. Rousset developed a plugin of Krakatoa for *Java Card programs*, used in smart cards industry. The goal was to handle all Java Card specific features, not present in pure Java. Indeed, the support of these features, provided by a native API, cannot be addressed directly in JML annotations but needs a special treatment. To this end, he modeled the Java Card memory heap by adapting the Java memory model of Krakatoa, in order to distinguish between persistent and volatile data. The transaction mechanism was formalized using Why specifications. Moreover, smart cards are sensitive to card tears, and it is an issue to ensure data consistency when the card is powered again. A way to express a postcondition a method must fulfill in case of a card tear was introduced in annotations. The exception mechanism of Why is then used to reason about card tears and to generate verification conditions to prove that postcondition. It is known through the PIN code check example that the combination of the use of transactions and the possibility of card tears can lead to a subtle security hole. As a proof of concept of our approach, we were able to automatically discover this threat in a naive implementation, and to prove safe a corrected version. This work has been published at the 4th IEEE International Conference on Software Engineering and Formal Methods [30].

6.2.6. Algebraic models for Java

C. Marché and N. Rousset also 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. This is work in progress, a preliminary version has been presented at the European Science Foundation Exploratory Workshop "Challenges in Java Program Verification" http://www.cs.ru.nl/~woj/esfws06/.

6.3. Automated Deduction

6.3.1. Decision Procedures for program proof

Participants: Malgorzata Biernacka, Sylvain Conchon, Evelyne Contejean, Jean-François Couchot, Johannes Kanig, Stéphane Lescuyer.

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, Yices, CVC lite, haRVey, RV-sat), 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 sort predicates transformation which was used till then. S. Lescuyer worked on this topic during his master thesis [41]: he has shown the correctness and the completeness of this approach. Moreover he implemented his work in the Why tool and get as a result that the new encoding is much more efficient than the predicate transformation.

Since September 2006, Jean-François Couchot works on the extension of this approach for encoding PSL into a multi-sorted logic in order to use more provers such as Yices. A key issue is to preserve the efficiency, especially for the arithmetic decision procedure.

However, 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's why S. Conchon and É. Contejean decided to develop a new tool called Ergo [40] 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 congruence closure algorithm parameterized by an equational theory X. Currently, $CC(X)$ can be instantiated by the empty equational theory and by the linear arithmetics. 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.

6.3.2. Data structures for decision procedures

Ergo uses the technique of hash-consing to share values that are structurally equal. Beyond the obvious advantage of saving memory blocks, hash-consing may also be used to speed up fundamental operations and data structures by several orders of magnitude when sharing is maximal. S. Conchon and J.-C. Filliâtre designed and implemented an Ocaml hash-consing library that encapsulates hash-consed terms in an abstract datatype, thus safely ensuring maximal sharing. This library is also parameterized by an equality that allows the user to identify terms according to an arbitrary equivalence relation. This work was presented at the SIGPLAN Workshop on ML 2006 [26].

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 which uses $CC(X)$, 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 will be presented at JFLA 2007 [27] .

6.3.3. Automated proofs and certificates of rewriting properties

In order to benefit from the full automation of automated theorem provers in interactive proof assistants, the former should not only provide a yes/no answer to a query, but return additional information (a proof trace) that permits the proof assistant to reconstruct the proof for the query. J. Kanig is currently working on extending the Ergo theorem prover to output such traces.

E. 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 [53] is that the Coq development is paramterized 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.

6.3.4. Termination proof

As the previous edition, C. Marché organized the third "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 2006 [31].

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 [16], and has been implemented into a new tool called MTT (http://www.lcc.uma.es/~duran/MTT/).

E. 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. 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).

6.4. Synchronous Programming

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

6.4.1. Data-flow models and automata models

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. In [25], we propose an extension of a synchronous data-flow language such as Lustre with imperative features expressed in terms of powerful state machine *à la* SyncChart. This extension is fully conservative in the sense that all the programs from the basic language still make sense in the extended language and their semantics is preserved. From a syntactical point of view this extension consists in hierarchical state machines that may carry at each hierarchy level a bunch of equations. This proposition is an alternative to the joint use of Simulink and Stateflow but improves it by allowing a fine grain mix of both styles. Technically, we base the extension on the use of clocks, translating imperative constructs into well-clocked data-flow programs from the basic language. The proposed extension has been implemented in the ReLuC compiler of SCADE/Lustre and in the Lucid Synchrone compiler. This work has been done in collaboration with Jean-Louis Colaço and Bruno Pagano from Esterel-Technologies.

In [25], we pursue this effort by providing a direct *logical semantics* as well as two extensions: a general form of state machines called parameterized state machines and valued signals, as can be found in Esterel. Parameterized state machines greatly reduce the reliance on error-prone mechanisms such as shared memory in automaton-based programming. Signals provide a new way of programming with multi-rate data in synchronous data-flow languages.

6.4.2. Tools

During the period, a complete rewriting of the Lucid Synchrone compiler has been done. The new version (V3) is freely available at http://www.lri.fr/~pouzet/lucid-synchrone. The language now includes the extensions proposed in [52] and [25]. A presentation of the language is given in a book chapter of a collection dedicated to real-time systems [14]. A reference manual and tutorial introduction have been written [34].

An other activity concerns the development of ReactiveML, an extension of Objective Caml with reactive constructs [64]. During the period, the language has been used for programming and simulating routing protocols in ad-hoc and sensor networks [37] (in collaboration with LIP6 and VERIMAG).

6.4.3. Alternative synchronous models

In collaboration with Albert Cohen and Christine Eisenbeis (INRIA Alchemy), Marc Duranton (Philips Natlabs, Eindhoven), we have introduced 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 and relying of a generalization of the conventional clock calculus of synchronous languages. This new clock calculus is expressed by a type-system with a sub-typing relation. This work is described in [51] and [24] and is the subject of the PhD thesis of Florence Plateau.

Finally, we have also worked on the problem of automatic desynchronization of synchronous programs, in collaboration with Alain Girault (INRIA Rhône-Alpes) and Xavier Nicollin (VERIMAG) [18].

During its Master studies, Alexandre Bertails has studies compilation techniques for video applications written in data-flow languages such as StreamIt (developped at MIT) [38].

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. This research activity is the subject of the post-doc of D. Biernacki and the PhD thesis of A. Bertails. They already developed the type-checker and are currently working on the clock calculus.

7. Contracts and Grants with Industry

7.1. System@tic: PFC

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

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 Axalto, CEA-LIST, Dassault Aviation and Trusted Logic.

This project is funded by the french ministry of industry (FCE).

7.2. CIFRE grants

Participants: June Andronick, 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 June Andronick studied for her PhD thesis from 2002 to Mar 2006. She worked on specification and proofs of properties of an operating system embedded on a smart card using Caduceus. The challenge was to deal with real code. In particular it was necessary to handle low-level translations (casts, unions) as well as to express global security properties.

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.

CIFRE, Dassault-Systèmes Elodie Malgouyres has started here PhD thesis in march 2005 under the supervision of Marc Pouzet (LRI, Proval) and Daniel Weil (DS, Grenoble). She is working on the type system for the language LCM (Logic Control Modeler), a synchronous language for programming industrial systems (PLC) inside the Catia suite. The language provides a rich type system with automatic type inference, a module system and is integrated inside a graphical environment.

8. Other Grants and Activities

8.1. National initiatives

8.1.1. GECCOO

Participants: Sylvain Conchon, Évelyne Contejean, Jean-Christophe Filliâtre, Claude Marché, Christine Paulin.

This project is funded by "ACI Sécurité et Informatique". The coordinator is C. Paulin. http://geccoo.lri.fr/.

Partners: TFC group, LIFC, Besançon; CASSIS project, LORIA, Nancy; Everest project, INRIA Sophia-Antipolis; VASCO group, LSR, Grenoble.

The objective of this project is to propose new methods and tools for the development of object-oriented programs, with strong guarantees of security. The project specifically focuses on programs embedded in smart cards or terminals which are written in subsets of Java (for example Java Card).

8.1.2. AVERROES

Participants: Pierre Corbineau, Jean-Pierre Jouannaud, Christine Paulin, Weiwen Xu.

This research project (09/2002-03/2006) is funded by national network on software technology (RNTL).

Partners: France Télécom R&D (coordinator); CRIL Technology Systèmes Avancés; LaBRI, Bordeaux; LORIA, Nancy; PCRI (LRI, LIX, INRIA Futurs), Saclay; LSV, Cachan.

The goal of the project is the development of formal methods for verifying, in a secure way, various properties (functional, non-functional, quantitative) appearing in industrial context. More information can be obtained from http://www-verimag.imag.fr/AVERROES/.

8.1.3. CAT

Participants: Jean-Christophe Filliâtre, Claude Marché.

CAT (C Analysis Tools) is a RNTL project related to the verification of C programs.

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.

8.1.4. A3PAT

Participants: Sylvain Conchon, Evelyne Contejean.

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. 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 E. 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.

8.1.5. CerPAN

Participants: Sylvie Boldo, Jean-Christophe Filliâtre.

CerPAN (Certification de Programmes d’Analyse Numérique), is an ANR project. 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.

8.1.6. ALIDECS

Participants: Marc Pouzet, Florence Plateau.

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 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, Estonie, Serbie, 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.

8.3. Visits, researcher invitation

8.3.1. Visits

Marc Pouzet was invited at the University of Bamberg for a week by Prof. Mendler in july 2006.

É. Contejean visited Salvador Lucas at the Polytechnical University of Valencia (December 2006) in order to discuss on a possible collaboration for producing an intermediate language for termination problems statements as well as for termination proofs.

8.3.2. Invitations

Pr. Yih-Kuen Tsay from National Taiwanese University and Dr. Bow-Yaw Wang from Academia Sinica visited the project for two days in December 2006.

9. Dissemination

9.1. Interaction with the scientific community

9.1.1. 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. She participated to the jury of CR2 hiring competition at INRIA Rhone-Alpes and INRIA Rennes as well as the national DR2 hiring competition.

9.1.2. Collective responsibilities outside INRIA

C. Paulin and C. Marché (until Sep. 06) are members of the ”commission de spécialistes”, section 27, University Paris-Sud 11. E. Contejean and C. Marché (until Sep.06) are members of the ”commission de spécialistes”, section 27, ENS Cachan.

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

C. Paulin is an elected member of the board of directors of University Paris-Sud 11, since April 2006.

9.1.3. Event organisation

C. Marché organized the 3rd ”Termination Competition” in June 2006 http://www.lri.fr/~marche/termination-competition/

9.1.4. 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/.

9.1.5. 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.

9.1.6. Program committees

C. Marché was a member of the program committee of 17th International Conference on Rewriting Techniques and Applications (RTA'06), the 8th International Workshop on Termination (WST'06) and the Workshop on Verified Software: Theories, Tools, and Experiments (VSTTE'06); both three held at the 3rd Federated Logic Conference (FLoC, Seattle, USA, August 2006).

C. Paulin was a member of the program committee of 3rd International Joint Conference on Automated Reasoning (IJCAR'06, Seattle, USA) and Journées Francophones des Langages Applicatifs (JFLA'06, Pauillac, France). She is participating to the program committees 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 will also 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 is a member of the program committee of the 20th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2007, Kaiserslautern, Germany).

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 2005 and 2006.

9.1.7. PhD juries

J.-C. Filliâtre was member of the PhD jury of Julien Signoles (University Paris-Sud 11, July 11th, 2006).

C. Marché was member of the PhD jury of Julien Narboux (University Paris-Sud 11, Sep. 26th, 2006). He reviewed the PhD manuscript of Mariela Pavlova (University of Nice, January 19th, 2007).

C. Paulin was member of the PhD jury of June Andronick (University Paris-Sud 11, March 29, 2006), Julien Signoles (University Paris-Sud 11, July 11th, 2006), Nicolas Oury (University Paris-Sud 11, September 15th, 2006, Cédric Miachon (University Paris-Sud 11, December 13th, 2006), Dan Hernest (Ecole Polytechnique and University Munich, December 14th, 2006). She reviewed the manuscript and was member of the PhD jury of Frédéric Dadeau (University of Franche-Comté, July 18th, 2006), Guillaume Melquiond (Ecole Normale Supérieure Lyon, November 21st, 2006) and Sabrina Tarento (University Nice Sophia-Antipolis, December 6th, 2006).

M. Pouzet was member of the Phd jury of Louis Mandel (University of Paris 6, may 2006). He reviewed the manuscript and was member of the PhD jury of Adrian Curic in october 2005 (University of Grenoble) and Philippe Dumont (University of Lille, 11th december 2005).

9.2. Teaching

9.2.1. Supervision of PhDs and internships

S. Conchon and E. Contejean supervised the internship of Stéphane Lescuyer (2nd year of the *Master Parisien de Recherche en Informatique*, feb-jul 2006) and Johannes Kanig (2nd year of the *Master Univ. Dresden, Germany*, sep 2006-feb 2007).

J.-C. Filliâtre supervised the internship of M. Dogguy (improvement of Why's graphical interface, January to May 2006).

M. Pouzet supervised the internship of C. Avenel (January to May 2006).

J.-C. Filliâtre supervised the Ph.D. of J. Signoles (addition of refinement to the ML language, defended in July 2006).

C. Paulin supervised the Ph.D. of Nicolas Oury (equalities and pattern matching with dependent types in the Calculus of Inductive Constructions, defended in Sept. 2006); and June Andronick (Formal verification of programs on smartcards, CIFRE co-supervised by B. Chetali at Gemalto company, defended March 2006)

C. Paulin supervises the Ph. D. thesis of Matthieu Sozeau (programming with dependent types in Coq) and Wendi Urribarí (towards certified libraries).

C. Marché supervises PhD of Thierry Hubert (CIFRE co-supervised by E .Ledinot, Dassault aviation), Nicolas Rousset (CIFRE co-supervised by B. Chetali, Gemalto) and Yannick Moy (CIFRE co-supervised by P. Crégut, France Télécom R&D).

M. Pouzet supervised the PhD of Louis Mandel, defended in May 2006. 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), of Alexandre Bertails (with Christine Paulin) and of Elodie Malgouyres (CIFRE, co-supervised by Daniel Weil, Dassault-Système Grenoble).

9.2.2. Graduate courses

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/
In 2005-2006 and 2006-2007, J.-C. Filliâtre is teaching part of the course on “Constructive Proofs” (12h per year).

In 2006-2007, É. Contejean lectured on advanced rewriting (15h) and S. Conchon lectured on decision procedures (6h) in the course on “Automated Deduction”.

Marc Pouzet is responsible of the course “Systèmes Synchrones” (24h). The course is shared with Jean Vuillemin (ENS) and Gérard Berry.

Other Master programs Marc Pouzet is responsible of a course (24h) on synchronous programming in the Professional Master (ISIC, “Ingenieurie des Systèmes Industriels Complexes” of Ecole Polytechnique, University Paris-Sud 11 and INSTN).

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

9.2.3. Other Courses

S. Conchon and C. Marché (until August 2006) and M. Pouzet are teaching as part of their duty (192h per year) at University Paris-Sud 11.

In 2005-2006, C. Marché was responsible for the first year of the master in Computer Science.

In 2005-2006 and 2006-2007, M. Pouzet is responsible for the second year of the master in Computer Science, for the “professional” branch.

J.-C. Filliâtre is teaching a course on compilers at university Paris-Sud 11 for undergraduate students (4th year).

9.3. Popularization

F. Plateau and Y. Moy prepared and animated an activity related to logics for the event “fête de la science” organized by the French ministry of Education and Research (October 13-14, 2006). 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] 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.

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

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

Year Publications

Doctoral dissertations and Habilitation theses

[11] J. ANDRONICK. *Modélisation et vérification formelles de systèmes embarqués dans les cartes à microprocesseur. Plateforme Java Card et Système d'exploitation*, Thèse de Doctorat, Université Paris-Sud, March 2006.

[12] N. OURY. *Egalité et filtrage avec types dépendants dans le calcul des constructions inductives*, Thèse de Doctorat, Université Paris-Sud, September 2006.

[13] J. SIGNOLES. *Extension de ML avec raffinement: syntaxe, sémantiques et système de types*, Thèse de Doctorat, Université Paris-Sud, July 2006.

Articles in refereed journals and book chapters

[14] P. CASPI, G. HAMON, M. POUZET. *Systèmes Temps-réel : Techniques de Description et de Vérification – Théorie et Outils*, N. NAVET (editor). , vol. 1, chap. Lucid Synchrone, un langage de programmation des systèmes réactifs, Hermes, 2006, p. 217-260.

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

[16] 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.

[17] 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.

[18] A. GIRAULT, X. NICOLLIN, M. POUZET. *Automatic Rate Desynchronization of Embedded Reactive Programs*, in "ACM Transactions on Embedded Computing Systems (TECS)", vol. 5, n^{o} 3, 2006.

Publications in Conferences and Workshops

[19] J. ANDRONICK. *Formally Proved Anti-tearing Properties of Embedded C Code*, in "2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation", November 2006.

[20] P. AUDEBAUD, C. PAULIN-MOHRING. *Proofs of Randomized Algorithms in Coq*, in "Mathematics of Program Construction, MPC 2006, Kuressaare, Estonia", T. UUSTALU (editor). , Lecture Notes in Computer Science, vol. 4014, Springer-Verlag, July 2006.

[21] 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.

[22] S. BOLDO, M. DAUMAS, W. KAHAN, G. MELQUIOND. *Proof and certification for an accurate discriminant*, in "12th IMACS-GAMM International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg,Germany", sep 2006.

[23] S. BOLDO, C. MUÑOZ. *Provably Faithful Evaluation of Polynomials*, in "Proceedings of the 21st Annual ACM Symposium on Applied Computing, Dijon, France", April 2006.

[24] 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.

[25] 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.

[26] S. CONCHON, J.-C. FILLIÂTRE. *Type-Safe Modular Hash-Consing*, in "ACM SIGPLAN Workshop on ML, Portland, Oregon", September 2006, http://www.lri.fr/~filliatr/ftp/publis/hash-consing2.ps.gz.

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

[28] J.-C. FILLIÂTRE. *Backtracking iterators*, in "ACM SIGPLAN Workshop on ML, Portland, Oregon", September 2006, http://www.lri.fr/~filliatr/publis/enum2.ps.gz.

[29] J.-C. FILLIÂTRE. *Itérer avec persistance*, in "Dix-septièmes Journées Francophones des Langages Applicatifs", INRIA, January 2006, http://www.lri.fr/~filliatr/ftp/publis/enum.ps.gz.

[30] 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/.

[31] C. MARCHÉ, H. ZANTEMA. *The Termination Competition 2006*, in "Extended Abstracts of the 8th International Workshop on Termination, WST'06", A. GESER, H. SONDERGAARD (editors). , August 2006, http://www.lri.fr/~marche/termination-competition/.

[32] M. SOZEAU. *Subset coercions in Coq*, in "post-workshop proceedings of TYPES'2006", to appear, 2007, http://www.lri.fr/~sozeau/research/russell/article.pdf.

Internal Reports

[33] C. PAULIN-MOHRING. *A library for reasoning on randomized algorithms in Coq*, Description of a Coq contribution, Université Paris Sud, January 2006, http://www.lri.fr/~paulin/ALEA/library.pdf.

[34] M. POUZET. *Lucid Synchrone, version 3. Tutorial and reference manual*, Université Paris-Sud, LRI, April 2006, http://www.lri.fr/~pouzet/lucid-synchrone.

[35] J. SIGNOLES. *Towards a ML extension with Refinement: a Semantic Issue*, Technical report, n^{o} 1440, LRI, University of Paris Sud, March 2006, http://www.lri.fr/~signoles/publis/signoles06semrfn.pdf.

[36] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual – Version V8.1*, July 2006, http://coq.inria.fr.

Miscellaneous

[37] F. BENBADIS, L. MANDEL, M. POUZET, L. SAMPER. *Simulation of Ad hoc Networks in ReactiveML*, Submitted to publication, June 2006.

[38] A. BERTAILS. *Langages synchrones avec horloges périodiques*, Technical report, Master Parisien de Recherche en Informatique, September 2006, http://www.lri.fr/~bertails/misc/master_thesis.pdf.

[39] S. BOLDO, J.-C. FILLIÂTRE. *Formal Verification of Floating-Point Programs*, Submitted, 2006.

[40] S. CONCHON, E. CONTEJEAN. *The Ergo automatic Theorem Prover*, 2006, http://ergo.lri.fr/.

[41] 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.

[42] N. OURY. *Pattern matching covering with dependent types using set approximations*, Submitted, 2006.

References in notes

[43] L. BACHMAIR (editor). *11th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 1833, Springer-Verlag, Norwich, UK, July 2000.

[44] H. GEUVERS, F. WIEDIJK (editors). *Types for Proofs and Programs Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 2 4-28, 2002, Selected Papers*, Lecture Notes in Computer Science, vol. 2646, Springer, 2003.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

[53] E. 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.

[54] 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.

[55] J.-C. FILLIÂTRE. *The Why verification tool*, 2005, http://why.lri.fr/.

[56] J.-C. FILLIÂTRE, T. HUBERT, C. MARCHÉ. *The Caduceus tool for the verification of C programs*, 2005, http://why.lri.fr/caduceus/.

[57] J.-C. FILLIÂTRE, C. MARCHÉ, C. PAULIN-MOHRING, X. URBAIN. *The* KRAKATOA *proof tool*, 2005, http://krakatoa.lri.fr/.

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

[59] 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.

[60] 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.

[61] 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.

[62] P. LETOUZEY. *A New Extraction for Coq*, in "TYPES 2002", H. GEUVERS, F. WIEDIJK (editors). , Lecture Notes in Computer Science, vol. 2646, Springer, 2003, http://www.lri.fr/~letouzey/download/extraction2002.ps.gz.

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

[64] 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.

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

[66] A. MINÉ. *The Octagon Abstract Domain*, in "AST 2001 in WCRE 2001", IEEE, IEEE CS Press, October 2001, p. 310–319, http://www.di.ens.fr/~mine/publi/article-mine-ast01.pdf.

[67] 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/.

[68] 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.