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 is a PCRI project common to INRIA-Futurs, CNRS, École Polytechnique and Université Paris XI. It involves researchers from LRI (Laboratoire de Recherche en Informatique, UMR 8623) and LIX (Laboratoire d'Informatique de l'École Polytechnique, UMR 7161).

Team Leader
Christine Paulin [Professor, Université Paris Sud, secondment INRIA]

Team Vice-Leader
Claude Marché [Assistant Professor, Université Paris Sud]

Administrative assistant
Marie-Carol Lopes [TR INRIA]

Research Scientist
Sylvie Boldo [CR INRIA, from 09/05]
Évelyne Contejean [CR CNRS, 50%]
Jean-Christophe Filliâtre [CR CNRS]
Jean-Pierre Jouannaud [Professor, Université Paris Sud, 10%]

Faculty Member
Sylvain Conchon [Assistant Professor, Université Paris Sud]

Ph. D. student
June Andronick [CIFRE Axalto]
Pierre Corbineau [MENRT, until 09/05]
Thierry Hubert [CIFRE Dassault]
Nicolas Rousset [CIFRE Axalto]
Matthieu Sozeau [MENRT, from 10/05]
Julien Signoles [MENRT]

Post-doctoral fellow
Weiwen Xu [Egide, France Telecom contract, until 02/05]

Student intern
Nicolas Ayache [Master MPRI, apr-sep 05]
Julien Roussel [Epita, feb-sep 05]
Matthieu Sozeau [Master MPRI, apr-sep 05]

2. Overall Objectives

2.1. Overall Objectives

2.1.1. From LogiCal to Proval

The members of the Proval project were part of the LogiCal project. The LogiCal project is carrying research in order to build *proof assistants* and is developing the Coq tool. Proof assistants are systems that can check that a proof is correct, they can help users build demonstrations interactively or automatically, store them in libraries, or export them towards other systems.

One possible application of proof assistants is the development of proof of programs. The constructive type theory underlying the Coq system is especially well-suited for developing certified functional programs. However, new directions of research appeared in the group for doing proofs of programs in a more general setting. We are also interested in dealing with programs written in programming languages like Java or C, with specification written in domain-specific languages and a general mechanism for generating proof obligations which can be discharged by different automatic or interactive proof tools.

A subgroup of the LogiCal project decided to propose the new Proval project which was recognised by INRIA in July 2004 and officially created in September 2005.

2.1.2. Proval approach

The Proval team studies advanced techniques for automated or interactive computer-assisted theorem proving and their application to the certification of programs.

The project develops a generic environment (Why) for proving programs; Why generates sufficient conditions for the program to meet its expected behaviour 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.

We are working on the following subjects:

Models and methods models of programming and specification languages adapted to critical software and suitable for computer-assisted proofs.

Architecture of systems generation and propagation of proof annotations, efficiency of proof obligations generation.

Automated deduction developing deduction algorithms for program verification, proof of termination, integration of interactive provers and automatic proof techniques.

3. Scientific Foundations

3.1. Automated deduction

Keywords: *decision procedure*, *proof traces*, *rewriting*, *termination*.

Participants: Sylvain Conchon, Evelyne Contejean, Pierre Corbineau, 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* [46], and further generalisations [6]. These new criteria and methods have been implemented into the CiME2 rewrite toolbox [15]. 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 organised 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 [8] and then to membership equational programs [40], a paradigm used in the *Maude* system [38].

Our research related to combination of decision procedures was initiated by a result [42] 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 [14], [17].

We are now focusing on the integration of decision procedures within user-assisted proof environments: in particular in the Coq system [10].

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.

A first experiment in that direction was C. Alvarado's thesis which designed a collaboration between Coq and the rewriting engine Elan [35]. We are currently experimenting with this idea in the domain of termination and of decision procedures. This is also the main purpose of the starting 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.

3.2. Proofs of programs

Keywords: *floating-point programs*, *proof obligations*, *specification*, *verification*.

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

Our group has been conducting research related to program verification for years. We mainly focus now on the verification of behavioural specifications for programming languages such as C, Java and ML.

In 1999, J.-C. Filliâtre introduced a new technique for the verification of imperative programs based on a functional translation into type theory [3]. A first implementation of this work allowed to verify imperative programs within the Coq proof assistant [16]. However it was quickly noticed that this approach was actually independent of the Coq system, and the design and implementation of the new tool "Why" started in 2001 [41]. This tool is a verification conditions generator: from an annotated program written in a small imperative language with Hoare logic-like specification, it generates conditions expressing the correctness and termination of the program. These verification conditions can be generated for several existing provers, including interactive proof assistants (Coq, PVS, HOL Light, Mizar) and automatic provers (Simplify, haRVey, CVC Lite). This multi-prover architecture is a powerful feature of Why: it spreads this technology well beyond the Coq community.

3.2.1. Proof of Java and C programs

In 2001, the team started considering the verification of Java programs. We designed a method to verify Java source code annotated with the Java Modeling Language (JML) and implemented it within a tool called Krakatoa [31]. The main challenge was the design of a suitable model for the Java memory heap in order to tackle programs with possible aliases (even if Java programs do not manipulate pointers explicitly, objects are passed around through pointers). Indeed, the Why tool does not handle aliases by itself. The Krakatoa model is inspired by old ideas from Burstall, recently used by Bornat [37] and others to verify pointers programs. The key idea is to separate the heap according to objects fields, since it is statically known that two different fields can not be aliased. Thanks to its modular architecture, the Why tool can be reused for the verification conditions generation. The memory heap model is declared as a set of abstract types, functions and predicates and Java code is then translated into Why input language. This original modular design has many advantages. First, the Krakatoa tool was implemented and released in less than one year. Second, the Krakatoa tool benefits from the multi-provers output of the Why tool, offering a wide range of proof systems for the verification of Java programs. The Krakatoa tool was successfully used for the formal verification of a commercial smart card applet [5] proposed by SchlumbergerSema company, this case study have been conducted in collaboration with B. Jacobs' group at University of Nijmegen and N. Rauch at University of Kaiserslautern. The Krakatoa tool is currently under use at Axalto (verification of other applets) and at the National Institute of Aerospace (avionics systems).

The success of Krakatoa and the constant request from industrial partners incited us to start a similar project for C programs in 2003. The whole design of Krakatoa could be reused, apart from the specification language that had to be designed from scratch (but it is greatly inspired by JML) and the memory model that had

to be refined to handle pointer arithmetic [4]. The resulted tool is called Caduceus. It is currently under experimentation at Dassault Aviation and Axalto.

3.2.2. Developing correct ML programs

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 [44], [45], 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 [2]. Other ML programs have been verified using similar techniques: recently in the Concert INRIA action, X. Leroy and others designed a formally certified optimising compiler from a subset of C to PowerPc assembly code [43].

We are also considering other ways to tackle ML programs. J. Signoles is working 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. Programs well-typed in this approach 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.

3.2.3. 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 behaviours (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 behaviour 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.

3.2.4. Timed automata

Among other programming paradigms, we collaborated to the definition of a model of timed automata in Coq [9]. It is integrated in the CALIFE platform, a general tool for specification and automatic or interactive verification of protocols which is developed within the context of the CALIFE and AVERROES projects.

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 controling 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 behaviour 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, under the LGPL licence, 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. E. Contejean has started to implement a saturation functor which can be instantiated by usual Knuth-Bendix completion, completion modulo AC, standard resolution and paramodulation. The ultimate goal is to use it in order to provide some traces for checking the proof when the goal (typically a proof obligation coming from the certification of a program) is reached.

5.2. The Why tool

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

Participant: Jean-Christophe Filliâtre [correspondent].

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 modeling language*, *Java programming language*, *JavaCard*.

Participants: Claude Marché [correspondent], Christine Paulin.

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 Axalto company, at National Institute of Aerospace (NIA) and DoCoMo labs.

5.4. The Caduceus tool

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

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

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 Axalto company and at Dassault Aviation company.

5.5. 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 2005) around 20000 web pages have been generated using Bibtex2html (+100% increase in one year).

6. New Results

6.1. Models and methods for proofs of programs

6.1.1. Functional programs and higher-order constructions

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

J. Signoles worked on an extension of ML with refinement. The aim of this PhD work is to add formal specifications in ML programs in order to prove them. He improved the syntax of the ML extension and its semantics foundations. He also finalised a type system generating proof obligations to verify with an external tool in order to ensure program correctness. This work has led to the implementation of a prototype and to first promising concrete experiments. Moreover he worked on practical programming and programming languages, in particular module systems *à la* ML [21], [28].

The combination of type theory and rewriting is also a powerful language for proofs and programs. N. Oury is working on the semantic of rewriting in the Calculus of Inductive Constructions (CIC). He proved the conservativity of extensionality over a slight extension of this calculus [27], this result justifies the fact that adding provable equalities as rewrite rules does not essentially change the theory. N. Oury is also studying the coverage problem of pattern matching with dependent types. The problem is undecidable in general, he proposed a new method for proving emptiness of certain patterns based on an approximation of the set of matching terms.

During his internship supervised by C. Paulin, M. Sozeau worked on an integration of predicate subtyping in Coq using coercions [33]. Predicate subtyping is a technique originating from the widely used PVS theorem prover which permits easier development of programs and proofs in a dependently typed language. The goal of his PhD work is to extend on this work and build a usable environment for programming with dependent types in Coq.

In parallel to his activities related to program verification, J.-C. Filliâtre is also conducting research in the framework of functional programming. He studied the use of persistent data structures to implement backtracking iterators [23]. S. Conchon, J.-C. Filliâtre and J. Signoles developed together a graph library for Objective Caml [21], which demonstrates the use of modules and functors to perform generic programming.

6.1.2. Models for Java and C programs with pointers

Participants: June Andronick, Thierry Hubert, Claude Marché, Christine Paulin, Nicolas Rousset.

Proofs of programs are based on a mathematical description of programs behaviour. We try to use models suitable for verification. For that purpose, we structure the memory into different mathematical objects. In the paper [26], we presented the model implemented in the current version of Krakatoa which is based on a separation according to object fields. This model was also designed with two different views: a first-order axiomatic theory suitable for reasoning with automatic tools like Simplify and also a set of higher-order definitions in the Coq system which justifies the consistency of the approach.

T. Hubert and C. Marché proposed a refinement of this method for Caduceus programs. An extra parameter is associated to each type of memory, which intuitively corresponds to a particular zone. The type-checking algorithm, using unification of type variables, can infer a more general type where program variables which are never aliased correspond to different zone parameters and can consequently be represented with different mathematical objects.

A separation model based on type analysis like the one in Caduceus is not compatible with certain type confusions which are done in a language like C using casts or union types. J. Andronick studied different techniques for modeling particular cases of theses constructions which are common practice in the smart-card operating systems.

In the JavaCard language, the memory is divided in two different parts: the persistent memory that is stored during the whole card lifetime, and the transient (i.e. volatile) memory that is cleared after each communication session between the card and the outside. A transaction mechanism is available in order to make several updates to persistent objects atomically even in the case of card tear in the middle of the operation. N. Rousset proposed a method for modeling properly this mechanism. The modeling he uses is an optimistic one: it doesn't do anything special in case of the commit of a transaction. When the transaction is aborted, programmatically or because of a card tear, the persistent memory is restored in its pre-transaction state. He adapted the memory model of Krakatoa to handle the separation of memory, and to be able to store a copy of the visible variables when entering a transaction sequence. Because the abort function potentially modifies all variables stored in the heap, a specific instance of it is generated for each JavaCard program.

6.1.3. Automatic complexity analysis of programs extracted from Coq proofs

Participants: Jean-Pierre Jouannaud, Weiwen Xu.

J.-P. Jouannaud and W. Xu studied an automatic complexity analysis for MiniML programs extracted from Coq proofs. The framework is similar to the one developed in the NUPRL project by R. Benzinger [36]. First, the operational semantics is enriched with annotations describing the number of steps taken by a program at runtime. Second, this semantics is generalised to programs with free parameters, yielding a formal description of the complexity of a program fragment, in terms of its time complexity. Third, recurrence relations are extracted from these descriptions allowing to analyse the asymptotic complexity. Programs can be higher-order, and admit an arbitrary number of inputs. An implementation is available for simple programs on natural numbers and lists. An abstract [25] has been published in the CLASE workshop.

6.2. Architecture of environments for proofs of programs

Participants: Nicolas Ayache, Jean-Christophe Filliâtre, Thierry Hubert, Claude Marché, Julien Roussel.

6.2.1. Krakatoa

J. Roussel, student from EPITA doing an internship from January to June 2005, developed a new front-end to Krakatoa, to allow the reuse of existing JML tools. The communication between tools is done via XML files. This work has been presented at GECCOO project meetings.

6.2.2. Caduceus

J.-C. Filliâtre devoted most of his research activity to the development of a verification tool for C programs, Caduceus [30]. T. Hubert implemented several improvements to the Caduceus tool: support for the `switch` statement, normalisation phase to smoothly support the `&` address operator, and a new handling of so-called

separations of structures, fields of structures and global variables. These improvements have been essential for the current experiments done at Dassault Aviation.

6.2.3. Why

J.-C. Filliâtre improved the Why tool [32] which is the common back-end of the Krakatoa tool for Java programs and the Caduceus tool for C programs. The core of the tool has been deeply redesigned, with a new verification conditions generator which produces more numerous but simpler proof obligations. A graphical user interface to assist the user in the process of automatic verification is under development.

J.-C. Filliâtre also supervised the master's thesis internship of N. Ayache [29]. N. Ayache studied the collaboration of the Coq proof assistant with several automatic decision procedures. He designed a translation from the higher-order logic of Coq (the Calculus of Inductive Constructions) to a sorted first-order logic, and proved it correct. Then he implemented this translation in Coq together with tactics to call several decision procedures (Simplify, haRVey, CVC Lite, Zenon) from the Coq toplevel. Up to now, the results from the decision procedures are trusted i.e. the corresponding properties are turned into axioms. This work will be improved: first to enlarge the logic fragment that is translated (e.g. to handle polymorphism) and second to translate back the proofs returned by the decision procedures when available (e.g. with Zenon).

6.3. Automated deduction

6.3.1. Decision procedures for proof of programs

Participants: Sylvain Conchon, Evelyne Contejean, Pierre Corbineau, Claude Marché.

Implementing and proving efficient combination algorithms is still a challenge: the gap between theory level descriptions found in the literature and efficient implementations is important.

S. Conchon is interested in designing and implementing a proof-producing theorem prover for first-order logic that combines at its core three "little engines" of proof: a propositional satisfiability solver, a ground decision procedure for equality and disequality, and the omega test for Presburger arithmetics.

He has started to implement a prototype whose propositional loop is based on an efficient SAT solver that he formalised and proved correct in the Coq proof assistant. In its current form, his prover can handle equalities and disequalities (thanks to an incremental and persistent congruence closure algorithm) and it can also perform quantifier elimination using a pattern-matching modulo equality algorithm.

P. Corbineau worked on integrating decision procedures in the Coq system. The main difficulty comes from the fact that Coq is based on a very general framework of inductive definitions and is based on intuitionistic logic. Also Coq requires a proof-term to be produced.

P. Corbineau extended his reflexion framework to encompass multi-sorted first-order logic with equality. He also extended his congruence-closure tactic with the theory of free constructors, which corresponds to the semantics of Coq's inductive datatypes. He defended his PhD in September 2005 [12].

E. Contejean and P. Corbineau worked together on producing proof traces from Knuth-Bendix completion using reflexion. A trace of ordered completion is produced by CiME, and the corresponding proof is checked by Coq. This approach has been validated on the relevant part of the TPTP library (http://www.cs.miami.edu/~tptp/). This work has been presented at the international "Conference on Automated Deduction (CADE-20)" [22].

6.3.2. Termination

Participants: Evelyne Contejean, Claude Marché.

C. Marché and E. Contejean continued to study techniques for proving termination of rewriting. The long-term cooperation with A.-P. Tomàs from Porto, E. Contejean and C. Marché leaded to a long paper describing new approaches for solving constraints arising in search for termination orderings based on polynomial interpretations, which will appear soon in the Journal of Automated Reasoning, part of a special issue recent trends in termination techniques [15]. C. Marché organised the second "Termination Competition", in collaboration with Hans Zantema of University of Eindhoven http://www.lri.fr/~marche/termination-competition/.

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. Some new achievements have recently been published in Information Processing Letters [18].

6.4. Applications

Proving significant examples is a good way to validate our approach and to consolidate the tools.

6.4.1. Proof of Schorr-Waite algorithm

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

To illustrate the effectiveness of the Caduceus tool, T. Hubert et C. Marché performed a full verification of a C implementation of the Schorr-Waite algorithm, 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. This case study has been presented at the IEEE International Conference on Software Engineering and Formal Methods [24].

6.4.2. Smart cards

Participants: June Andronick, Claude Marché, Christine Paulin, Nicolas Rousset.

J. Andronick worked on the verification of C programs embedded on smart cards; this work was presented at the international symposium on Formal Methods (FM'05) [19]. Using the Caduceus/Why tool, she is working directly on code developed by engineers at Axalto. This code makes use of low-level constructions like bit arrays or casts which requires to adapt the Caduceus approach.

N. Rousset implemented in Krakatoa specialised techniques to support the JavaCard transaction mechanism. He experimented his approach on an industrial electronic purse applet named Payflex developed by Axalto.

6.4.3. Floating-point programs

Participant: Sylvie Boldo.

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 [39] and a new result about polynomial evaluation that is formally proved to be faithful on mild assumptions (met for example when computing elementary functions). This work will be presented at the ACM Symposium on Applied Computing [20].

7. Contracts and Grants with Industry

7.1. France Telecom

Participants: Jean-Pierre Jouannaud, Weiwen Xu.

This collaboration with France Télécom R&D concerns complexity analysis of embedded software. The contract started in October 2003 for 3 years.

7.2. 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.

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 JavaCard).

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 new 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: reusability of components;
- verification: threats detection (division by zero), fault propagation and proof of global properties;
- maintenance and evolution: 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 (http://coq.inria.fr) and the prover CiME. (http://cime.lri.fr).

8.1.5. CerPAN

Participants: Sylvie Boldo, Jean-Christophe Filliâtre.

CerPAN (Certification de Programmes d'Analyse Numérique), is a new 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: Université Paris XIII, INRIA, CNAM.

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

J.-C. Filliâtre visited Martin Berger at the Queen Mary University of London (May 2005), starting an informal collaboration related to the verification of ML programs.

S. Boldo presented her work at the "Algebraic and Numerical Algorithms and Computer-assisted Proofs" seminar in Dagstuhl, Germany in September 2005. She also gave a talk at the "Journées Arinews" in Perpignan in November 2005.

C. Paulin participated to the second Taiwanese-French conference in Information technologies in Tainan, Taiwan in March 2005. During this visit, she also gave a seminar at Academia Sinica in Taipei.

8.3.2. Invitations

Pr. Yih-Kuen Tsay from National Taiwanese University in Taipei visited the project for one week in July 2005.

Dr Jacek Chrząszcz and Daria Walukiewicz from Warsaw university visited the logiCal and Proval projects in June and July 2005.

9. Dissemination

9.1. Interaction with the scientific community

9.1.1. Collective responsibilities within INRIA

C. Paulin was chairman of the Section Locale d'Auditions (local hiring committee) for the INRIA Futurs CR2 hiring competition (approximately 60 candidates for 6 positions).

Since October 2005, C. Paulin is vice-president of the project committee of INRIA Futurs and member of the national evaluation committee of INRIA.

9.1.2. Collective responsibilities outside INRIA

C. Marché and C. Paulin are members of the ”commission de spécialistes”, section 27, Université Paris 11.
E. Contejean and C. Marché 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.

9.1.3. Event organisation

J.-C. Filliâtre and C. Paulin together with B. Werner from the LogiCal team edited the post-proceedings [11] containing selected papers from the TYPES'2004 workshop http://types2004.lri.fr, the annual workshop of the European TYPES coordination action.

C. Marché organised the 2nd ”Termination Competition” in April 2005 http://www.lri.fr/~marche/termination-competition/

9.1.4. Learned societies

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

9.1.5. Program committees

J.-C. Filliâtre was a member of the program committee of the Workshop on ML 2005 (Tallinn, Estonia, September 2005).

J.-C. Filliâtre and C. Paulin were members of the program committee of Theorem Proving in Higher Order Logics (TPHOLs, Oxford, UK, August 2005).

C. Paulin was a member of the program committee of 16th International Conference on Rewriting Techniques and Applications (RTA, Nara, Japan, April 2005) and of Constructive Logic for Automated Software Engineering Satellite event of ETAPS 2005 (CLASE, Edinburgh, Scotland, April 2005)

C. Paulin is 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).

9.1.6. PhD juries

C. Paulin participated in the PhD juries of Sabri Boughorbel (Université Paris Sud, 07/05) and David Pichardie (Université Rennes 1, 12/05). C. Paulin participated in the habilitation jury of Hugo Herbelin (Université Paris Sud, 12/05).

9.1.7. Habilitation

C. Marché defended his habilitation in December [13]. The members of the jury were G. Barthe (INRIA Sophia-Antipolis), J. Beauquier (Université Paris Sud), J. Giesl (Aachen university), C. Kirchner (INRIA Lorraine), R. K. Leino (Microsoft research) and P. Lescanne (ENS Lyon).

9.2. Teaching

9.2.1. Supervision of PhDs and internships

C. Marché supervised the 5th year-internship of the “Ecole Pour l'Informatique et les Techniques Avancées” of J. Roussel, from January to June 2005.

C. Paulin supervised M. Sozeau internship (Research Master, MPRI). J.-C. Filliâtre supervised N. Ayache internship (Research Master, MPRI).

J.-C. Filliâtre supervises the Ph.D. of J. Signoles (addition of refinement to the ML language).

C. Marché supervised the Ph.D. of P. Corbineau (proof automation in Coq, thesis defended Sept 30). He is currently supervising Thierry Hubert (CIFRE with Dassault aviation) and together with B. Chetali at Axalto, he supervises the thesis of Nicolas Rousset (CIFRE).

C. Paulin supervises the Ph. D. thesis of Matthieu Sozeau (programming with dependent types in Coq) and Nicolas Oury (equalities and pattern matching with dependent types in the Calculus of Inductive

Constructions); she is also co-advisor of June Andronick's thesis (Formal verification of programs on smartcards) together with B. Chetali at Axalto.

9.2.2. Graduate courses

Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/
In 2004-2005, J.-C. Filliâtre and C. Marché were respectively teaching part of the course on "Constructive Proofs" and "Automated Deduction".
In 2005-2006, J.-C. Filliâtre is teaching part of the course on "Constructive Proofs" (12h). E. Contejean lectured on advanced rewriting (6h) and S. Conchon lectured on decision procedures (4h) in the course on "Automated Deduction".

TYPES Summerschool During this school (Göteborg, Sweden, August 2005) J.-C. Filliâtre gave two lectures: "Program verification using Coq" and "Introduction to the WHY tool" and C. Paulin gave two lectures on the Krakatoa tool.

Graduate school Since sept 2005, 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, C. Marché and C. Paulin are teaching as part of their duty at University Paris Sud.

In 2004-2005 C. Paulin and C. Marché were responsible for the 3rd and 4th year of undergraduate studies in Computer Science at University Paris Sud. C. Paulin was also the coordinator of the master project. In 2005-2006, C. Marché is responsible for the first year of the master in Computer Science.

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

10. Bibliography

Major publications by the team in recent years

[1] E. CONTEJEAN. *A certified AC matching algorithm*, in "15th International Conference on Rewriting Techniques and Applications", V. VAN OOSTROM (editor). , Lecture Notes in Computer Science, vol. 3091, Springer-Verlag, June 2004, p. 70–84.

[2] 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.

[3] 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.

[4] J.-C. FILLIÂTRE, C. MARCHÉ. *Multi-Prover Verification of C Programs*, in "Sixth International Conference on Formal Engineering Methods, Seattle, USA", Lecture Notes in Computer Science, Springer-Verlag, November 2004.

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

[6] 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.

[7] 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.

[8] 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", L. BACHMAIR (editor). , Lecture Notes in Computer Science, vol. 1833, Springer-Verlag, July 2000, p. 270–273, http://bibiserv.techfak.uni-bielefeld.de/talp/.

[9] C. PAULIN-MOHRING. *Modelisation of Timed Automata in Coq*, in "Theoretical Aspects of Computer Software (TACS'2001)", N. KOBAYASHI, B. PIERCE (editors). , Lecture Notes in Computer Science, vol. 2215, Springer-Verlag, 2001, p. 298-315.

[10] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual – Version V8.0*, April 2004, http://coq.inria.fr.

Books and Monographs

[11] J.-C. FILLIÂTRE, C. PAULIN, B. WERNER (editors). *Types for proofs and Programs, International Workshop, TYPES 2004, Revised Selected Papers*, Lecture Notes in Computer Science, vol. 3839, Springer-Verlag, 2006.

Doctoral dissertations and Habilitation theses

[12] P. CORBINEAU. *Démonstration Automatique en Théorie des Types*, Thèse de Doctorat, Université Paris-Sud, September 2005.

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

Articles in refereed journals and book chapters

[14] S. CONCHON, S. KRSTIĆ. *Strategies for Combining Decision Procedures*, in "Theoretical Computer Science", to appear, 2005.

[15] E. CONTEJEAN, C. MARCHÉ, A. P. TOMÁS, X. URBAIN. *Mechanically proving termination using polynomial interpretations*, in "Journal of Automated Reasoning", 2005, http://www.lri.fr/~marche/contejean05jar.ps.

[16] J.-C. FILLIÂTRE. *Formal Proof of a Program: Find*, in "Science of Computer Programming", to appear, 2005, http://www.lri.fr/~filliatr/ftp/publis/find.ps.gz.

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

[18] S. LUCAS, C. MARCHÉ, J. MESEGUER. *Operational Termination of Conditional Term Rewriting Systems*, in "Information Processing Letters", 2005.

Publications in Conferences and Workshops

[19] J. ANDRONICK, B. CHETALI, C. PAULIN-MOHRING. *Formal Verification of Security Properties of Smart Card Embed ded 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.

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

[21] S. CONCHON, J.-C. FILLIÂTRE, J. SIGNOLES. *Le foncteur sonne toujours deux fois*, in "Journées Francophones des Langages Applicatifs", March 2005, http://www.lri.fr/~signoles/publis/cfs05jfla.ps.gz.

[22] 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.

[23] J.-C. FILLIÂTRE. *Itérer avec persistance*, in "Journées Francophones des Langages Applicatifs (JFLA'06)", to appear, January 2006, http://www.lri.fr/~filliatr/ftp/publis/enum.ps.gz.

[24] 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.

[25] J.-P. JOUANNAUD, W. XU. *Automatic Complexity Analysis for Programs extracted from Coq Proofs*, in "Constructive Logic for Automated Software Engineering (CLASE), Satellite event of ETAPS 2005, Edinburgh", April 2005.

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

[27] N. OURY. *Extensionality in the Calculus of Constructions*, in "18th International Conference on Theorem Proving in Higher Order Logics", J. HURD, T. MELHAM (editors). , Lecture Notes in Computer Science, vol. 3603, Springer-Verlag, August 2005.

[28] J. SIGNOLES. *Une approche fonctionnelle du modèle vue-contrôleur*, in "Journées Francophones des Langages Applicatifs", In French, March 2005, http://www.lri.fr/~signoles/publis/signoles05jfla.ps.gz.

Miscellaneous

[29] N. AYACHE. *Coopération d'outils de preuve interactifs et automatiques*, Master thesis, Université Paris 7, 2005.

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

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

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

[33] M. SOZEAU. *Coercion par prédicats en Coq*, Master thesis, Université Paris 7, 2005.

Bibliography in notes

[34] 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.

[35] C. ALVARADO. *Réflexion pour la réécriture dans le calcul des constructions inductives*, Ph. D. Thesis, Université Paris-Sud, December 2002.

[36] R. BENZINGER. *Automated Complexity Analysis of Nuprl Extracted Programs*, in "Journal of Functional Programming", 2001.

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

[38] 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.

[39] M. DAUMAS, L. RIDEAU, L. THÉRY. *A generic library of floating-point numbers and its application to exact computing*, in "14th International Conference on Theorem Proving in Higher Order Logics, Edinburgh, Scotland", 2001, p. 169-184, http://perso.ens-lyon.fr/marc.daumas/SoftArith/DauRidThe01.ps.

[40] 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.

[41] J.-C. FILLIÂTRE. *Why: a multi-language multi-prover verification tool*, Research Report, n^o 1366, LRI, March 2003, http://www.lri.fr/~filliatr/ftp/publis/why-tool.ps.gz.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.