INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team ProVal

Proof of programs

Futurs

THEME SYM

Activity Report

2004

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, FRE 2653).

Head of project-team
Christine Paulin [Professor, Université Paris Sud]

Vice-head of project team
Claude Marché [MCF, Université Paris Sud]

Administrative assistant
Marie-Carol Lopes [TR INRIA, 10/04-08/05]

CNRS staff
Jean-Christophe Filliâtre [CR]

Paris XI staff
Sylvain Conchon [MCF, Université Paris Sud]

External researchers
Jean-Pierre Jouannaud [Professor, Université Paris Sud]
Évelyne Contejean [CR CNRS]
Jean Duprat [MCF, ENS-Lyon]

Visiting researchers
Annabelle McIver [Professor, Macquarie University, Sydney, feb-july 04]
Alexander Flegontov [Professor, St Petersburg University, nov-dec 04]

Ph. D. student
June Andronick [CIFRE Axalto]
Pierre Corbineau [MENRT]
Thierry Hubert [CIFRE Dassault, from 11/04]
Pierre Letouzey [MENRT, until 08/04]
Nicolas Rousset [CIFRE Axalto, from 11/04]
Julien Signoles [MENRT]

Post-doctoral fellow
Weiwen Xu [Egide, France Telecom contract, until 12/04]

Student intern
Pierre Clairambault [ENS Lyon, may-july 04]
Thierry Hubert [DEA, april-sept 04]
Vikrant Chaudhary [IIT student, may-july 04]

2. 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 as an INRIA team in july 2004.

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 sufficent 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.

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.

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

3. Scientific Foundations

3.1. Automated deduction

Keywords: *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 decisions procedures, and generation of proof traces.

On termination topic, we have been interested in the design of new automatizable techniques. A fundamental result of ours is a new criterion for checking termination *modularly* and *incrementally* [37], and further generalizations [33]. These new criteria and methods have been implemented into the CiME2 rewrite toolbox [25]. 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 [34] and then to membership equational programs [8] paradigm used in the *Maude* system [24].

Our research related to combination of decision procedures was initiated by a result [30] 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 Krstic 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.

We are now focusing on the integration of decision procedures within user-assisted proof environments: in particular in the Coq system [13].

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 should give a *proof trace*, that is a kind a 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 [22]. We are currently experimenting with this idea in the domain of termination and of decision procedures.

3.2. Proofs of programs

Participants: Jean Duprat, Jean-Christophe Filliâtre, Jean-Pierre Jouannaud, Pierre Letouzey, Claude Marché, Nicolas Oury, Christine Paulin, Julien Signoles.

Our group has been conducting research related to program verification for years. We mainly focus on the verification of behavioral 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 [29]. A first implementation of this work allowed to verify imperative programs within the Coq proof assistant [28]. 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 [27]. 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 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 [5]. 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 [23] 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 [11] 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).

3.2.2. Proof of C programs

The success of Krakatoa and the constant request from industrial partners incited us to start a similar project for C programs on 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 [10]. The resulted tool is called Caduceus. It is currently under experimentation at Dassault Aviation and Axalto.

3.2.3. 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 [32] [1], 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 [9]. Other ML programs have been verified using similar techniques, including the Koda-Ruskey algorithm [31] or some exact real arithmetic algorithms [26]. During his master thesis, N. Oury investigated methods to improve the efficiency of extracted programs by a safe substitution of data structures at extraction time [35].

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 non-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 to use directly a powerful type theory as a programming language where dependent types are expressing the specification.

3.2.4. Timed automata

Among other programming paradigms, we collaborated to the definition of a model of timed automata in Coq [36]. 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. The current research includes the modeling and analysis of randomised algorithms.

4. Application Domains

4.1. Panorama

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

Many systems in telecommnication, 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 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 AerospaceNIA, Hampton, USA.

5. Software

5.1. The CiME rewrite toolbox

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

Participants: Claude Marché [correspondant], Evelyne Contejean.

CiME is a rewriting toolbox. Distributed since 1996 as open source, under the LGPL licence, at URL http://cime.lri.fr. Beyond its few tens of users, CiME is used as back-end of 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, and the CARIBOO tool (http://www.loria.fr/equipes/protheo/SOFTWARES/CARIBOO/) for termination of rewriting under strategies.

5.2. The Why tool

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

Participant: Jean-Christophe Filliâtre [correspondant].

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 [correspondant], 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 [correspondant], 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://why.lri.fr/caduceus. 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 [correspondant], 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 2004) 10800 web pages have been generated using Bibtex2html.

6. New Results

6.1. Models and methods for proofs of programs

6.1.1. Functional programs and higher-order constructions

Participants: Jean-Christophe Filliâtre, Pierre Letouzey, Nicolas Oury, Christine Paulin, Julien Signoles.

P. Letouzey defended his thesis [1] in July. This thesis described a new mechanism for automatically extracting Haskell or Caml programs from constructive proofs in Coq. His thesis contains a new proof of correctness based on a semantical approach. The work on ML typing of extracted programs allows the extraction of modular programs. It was possible to develop using extraction a certified version of Caml libraries for finite sets (which fixes two small errors in the balancing scheme), P. Letouzey presented this work at ESOP'04 [9].

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 has defined a wide-spectrum ML language in which it is possible to express programs and specifications. He has defined three differents semantics of this language and proven their equivalence [20]. Typing has been expressed in First Order Logic allowing generation of proof obligations to be verifed with an external tool. Morever he worked on practical programming and programming languages, in particular module systems *à la* ML[15][21].

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. He proved the conservativity of extensionality over a slight extension of this calculus. A report on this fact is being written. He is studying some heuristics to analyse the coverage of pattern matching, a indecidable problem in presence of dependent types.

6.1.2. Randomised programs

Participants: Pierre Clairambault, Annabelle McIver, Christine Paulin.

C. Paulin continued her work on developing a Coq library for modeling randomised algorithm using a monadic transformation. During his internship, P. Clairambault used this library in order to prove the functional and probabilistic properties of a randomised algorithm for finding a minimal cut in a graph. This work suggested new general proof rules [14] for estimating the probability of a program to satisfy a certain property.

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 Minimal programs extracted from Coq proofs. The framework is similar to the one used in the NUPRL project. First, the operational semantics is enriched with annotations describing the number of steps taken by a program at runtime. Second, this semantics is generalized 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 and a research report is in progress.

6.2. Architecture of environments for proofs of programs

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

Jean-Christophe Filliâtre devoted most of his research activity to the development of, Caduceus [16] in joint work with C. Marché and Th. Hubert. This tool reuses the main methodology developed for the Krakatoa tool for Java programs [17], but a specific model for C programs has been designed (pointer arithmetic introduces an additional difficulty w.r.t. Java programs). The Caduceus tool is already available and under experimentation in two industrial companies. This work has been presented at the ICFEM conference [10] (Seattle, November 2004).

J.-C. Filliâtre also improved the Why tool [18] which is the common back-end of the Krakatoa tool for Java programs [17] and the Caduceus tool for C programs [16]. Predicates definitions and axioms have been added. Why polymorphism is now supported in all provers, including those with no support for polymorphism such as PVS. A support for the CVC Lite decision procedure has been added, increasing the total number of supported provers to 7.

6.3. Automatic deduction

6.3.1. Decision procedures for proof of programs

Participants: Sylvain Conchon, Evelyne Contejean, Pierre Corbineau, Jean-Christophe Filliâtre, 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 still important.

S. Conchon, in collaboration with S. Krstic from Intel Strategic CAD Labs, presented a system of inference rules together with a strategy language that are fine grained enough to model most of the mechanisms and optimizations currently used in existing tools. This work [2] and its related work [3] contribute to the understanding of combination of decision procedures and allow much clearer proofs of correctness.

S. Conchon is also interested in designing and implementing first-order decision procedures that use SAT solvers to improve the performance of the propositional search of the overall procedure. He has started to formalize the optimizations found in efficient SAT solvers (backjumping and conflict-clause learning) and plans to prove the correctness of such solvers in Coq.

E. Contejean has started to implement inside CIME a saturation functor which can be instanciated 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.

P. Corbineau works 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 published a paper [7] about the proof of cut-elimination of a contraction-free sequent calculus for first-order intuitionnistic logic. To improve the performance of the `firstorder` tactic implemented in the latest distributed version of Coq, he is currently working on a backend of his procedure based on reflection. This approach already gave encouraging results in the propositionnal case. He also worked on extending his congruence-closure tactic with the theory of free constructors, which corresponds to the semantics of Coq's inductive datatypes.

6.3.2. Termination

Participants: Evelyne Contejean, Thierry Hubert, Claude Marché.

C. Marché continued to study techniques for proving termination of rewriting. A major contribution, a new criterion for checking termination *modularly* and *incrementally*, has been pulished in a journal [38], and furthermore a generalization of it to termination modulo associativity and commutativity [4]. In cooperation with A.-P. Tomàs from Porto, E. Contejean and C. Marché also proposed a new approach for solving constraints arising in search for termination orderings based on polynomial interpretations [12]. C. Marché organized the first ”Termination Competition”, held in conjunction with the 7th International Workshop on Termination (WST 2004), Aachen, Germany, June 1-2, 2004, http://www.lri.fr/~marche/wst2004-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 [8] paradigm used in the *Maude* system.

Th. Hubert designed a method to produce, from a termination proof based on dependency pairs and dependency graph criteria, a Coq script that certifies this proof [19]. This method has been implemented in CiME2: a proof trace, checkable by Coq, is generated from automated termination proofs.

6.4. Applications

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

6.4.1. Proof of JavaCard applets

Participants: Vikrant Chaudhary, Jean Duprat, Claude Marché.

Demoney is an applet for an electronic purse proposed by the company Trusted Logic in order to experiment with verification tools. It illustrates the main aspects of real applets. J. Duprat worked on the specification, he proposed a Coq modelisation of the expected behavior of the applet. His work was presented at the GECCOO meeting in May. V. Chaudhary, supervised by C. Marché studied the specification of the applet and the proof of one of the main methods. His work concentrates on a systematic method for expressing in JML specification bases on state transition. The example suggested improvements of the why and Krakatoa tools in order to deal with large case expressions.

6.4.2. Proof of Schorr-Waite algorithm

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

In october-november 2004, under supervision of C. Marché, Th. Hubert completed a formal proof 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 a benchmark for verification tools. A report is under preparation.

6.4.3. Smart cards

Participant: June Andronick.

J. Andronick worked on the verification of C programs embedded on smart cards [6]. She is 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.

7. Contracts and Grants with Industry

7.1. France Telecom

Participants: Jean-Pierre Jouannaud, Weiwen Xu.

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

7.2. ECVAES

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

This project received a label in the Oppidum program of the french government but is not yet funded.

Partners: Axalto, Esterel Technology, INRIA.

The goal of the project is the conception and validation of secure and safe embedded applications on smartcards. http://www.telecom.gouv.fr/programmes/oppidum/ecvaes.htm

8. Other Grants and Activities

8.1. National initiatives

8.1.1. GECCOO

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

This project is funded by ”ACI Sécurité et Informatique”. The coordinator and scientific director for LRI is C. Paulin. http://geccoo.lri.fr/.

Partners: LogiCal PCRI project, LRI, Orsay (coordinator); 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, Thierry Hubert, Christine Paulin, Weiwen Xu.

This research project is funded by national network on software technology (RNTL). http://www-verimag.imag.fr/AVERROES/

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 (fonctional, non-fonctional, quantitative) appearing in industrial context.

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 2 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 the National Institute of Aerospace in March 2004.

E. Contejean and C. Marché visited J. Meseguer's team in University of Illinois at Urbana-Champaign in july 2004. They are working together in the framework of a CNRS-NSF bilateral collaboration.

8.3.2. Invitations

Alexander Flegontov, professor at university St Petersburg, visited for 2 months from November 2004 in the framework of a jumelage between CNRS and Russian Academy of Science. He is working on coercion in type theory.

Annabelle McIver, professor at university Mc Quarie visited our team for 6 months. She is working on randomised algorithms.

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 (45 candidates for 6 positions).

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 participates to the steering committees of the Pluridisciplinary Thematic Networks RTP 19 SECC (Systèmes Embarqués Complexes ou Contraints) and RTP 23 (Méthodes Mathématiques de l'Informatique).

9.1.3. Event organisation

J.-C. Filliâtre and C. Paulin are participating to the organisation of the TYPES'2004 workshop http://types2004.lri.fr, the annual workshop of the european TYPES coordination action (more than 100 participants).

C. Marché organised the 1st ”Termination Competition”, held in conjunction with the 7th International Workshop on Termination (WST 2004), Aachen, Germany, June 1-2, 2004, http://www.lri.fr/~marche/wst2004-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

C. Marché was a member of the International Workshop on Termination (WST 2004), Aachen, Germany, June 1-2, 2004

C. Paulin and J.-C. Filliâtre are members of the program committee of Theorem Proving in Higher Order Logics (TPHOLs 2004), Park City, Utah, USA, September 14-17, 2004. They are also participating to the program committee of TPHOLs 2005.

C. Paulin is member of the program committee of 16th International Conference on Rewriting Techniques and Applications RTA 2005.

9.1.6. PhD juries

Christine Paulin was reviewer of David Chemouil's PhD thesis (University of Toulouse, september 2003), of Luis Felipe-Cruz thesis at (Nijmegen University-Holland, June 04). She participated in the PhD jury of Jacek Chrząszcz (University Paris Sud-University of Warsaw, January 2004)

9.2. Teaching

9.2.1. Supervision of PhDs and internships

Claude Marché supervised Vikrant Chaudhary summer internship (IIT student). He also supervised the DEA internships of Thierry Hubert.

Christine Paulin supervised Pierre Clairambault internship (ENS Lyon, 1st year) on proofs of probabilistic programs in Coq.

Jean-Christophe Filliâtre supervises the Ph.D. of Julien Signoles (addition of refinement to the ML language).

Claude Marché supervises the Ph.D. of Pierre Corbineau (proof automation in Coq), Thierry Hubert (CIFRE with Dassault aviation) and together with B. Chetali at Axalto, he supervises the thesis of Nicolas Rousset (CIFRE).

Christine Paulin was the thesis advisor of Pierre Letouzey (thesis defended July 9th). She supervises the Ph. D. thesis of Nicolas Oury and is also co-advisor of June Andronick's thesis (CIFRE) at Axalto.

9.2.2. Graduate courses

DEA Programmation, Sémantique, Preuves,et Langages (2003-2004) Jean-Christophe Filliâtre and Christine Paulin participated to the optional course on "Calculus of Inductive Constructions"; Claude Marché participated to the optional course on "Termination"

Master Parisien de Recheche en Informatique (2004-2005) Jean-Christophe Filliâtre and Claude Marché are respectively teaching part of the course on "Constructive Proofs" and "Automated Deduction".

9.2.3. Other Courses

Sylvain Conchon, Claude Marché and Christine Paulin are teaching as part of their duty at University Paris Sud. Christine Paulin and Claude Marché are responsible for the 3rd and 4th year of undergraduate studies in Computer Science at University Paris Sud. Christine Paulin is also the coordinator of the master project.

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

10. Bibliography

Doctoral dissertations and Habilitation theses

[1] 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.

Articles in referred journals and book chapters

[2] S. CONCHON, S. KRSTIĆ. *Strategies for Combining Decision Procedures*, in "Theoretical Computer Science", 2004.

[3] S. KRSTIĆ, S. CONCHON. *Canonization for Disjoint Unions of Theories*, in "Information and Computation", Special Issue of Information and Computation dedicated to a refereed selection of papers presented at CADE-19, 2004.

[4] 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.

[5] 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.

Publications in Conferences and Workshops

[6] J. ANDRONICK, B. CHETALI. *Proving the Source Code of a Smart Card Operating System using Formal Methods*, in "e-SMART'04 Proceedings", september 2004.

[7] P. CORBINEAU. *First-order reasoning in the calculus of inductive constructions*, in "3rd International Workshop on Types for Proofs and Programs, Torino, Italy", S. BERARDI, M. COPPO, F. DAMIANI (editors)., Lecture Notes in Computer Science, vol. 3085, Springer-Verlag, April 2004, p. 162–177.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

Internal Reports

[12] E. CONTEJEAN, C. MARCHÉ, A.-P. TOMÁS, X. URBAIN. *Mechanically proving termination using polynomial interpretations*, Research Report, n^o 1382, LRI, 2004, http://www.lri.fr/~urbain/textes/rr1382.ps.gz.

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

Miscellaneous

[14] P. CLAIRAMBAULT. *Preuve de programmes probabilistes en Coq*, Rapport de stage, September 2004.

[15] S. CONCHON, J.-C. FILLIÂTRE, J. SIGNOLES. *Le foncteur sonne toujours deux fois*, submitted, 2004.

[16] J.-C. FILLIÂTRE, C. MARCHÉ. *The Caduceus tool for the verification of C programs*, 2004, http://why.lri.fr/caduceus/.

[17] J.-C. FILLIÂTRE, C. MARCHÉ, C. PAULIN, X. URBAIN. *The* KRAKATOA *proof tool*, 2004, http://krakatoa.lri.fr/.

[18] J.-C. FILLIÂTRE. *The Why verification tool*, 2004, http://why.lri.fr/.

[19] T. HUBERT. *Certification des preuves de terminaison en Coq*, In French, Rapport de DEA, Université Paris-Sud, September 2004.

[20] J. SIGNOLES. *Towards a ML Extension with Refinement: a Semantic Issue*, submitted, 2004.

[21] J. SIGNOLES. *Une approche fonctionnelle du "Modèle-Vue-Contrôleur"*, submitted, 2004.

Bibliography in notes

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

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

[24] 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.

[25] E. CONTEJEAN, C. MARCHÉ, B. MONATE, X. URBAIN. *Proving Termination of Rewriting with* C*i*ME, in "Extended Abstracts of the 6th International Workshop on Termination, WST'03", A. RUBIO (editor)., Technical Report DSIC II/15/03, Universidad Politécnica de Valencia, S pain, June 2003, p. 71–73, http://cime.lri.fr.

[26] J. CRECI. *Certification d'algorithmes d'arithmétique réelle exacte dans le système Coq*, Rapport de DEA, Université Paris-Sud, Orsay, France, September 2002.

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

[28] J.-C. FILLIÂTRE. *Formal Proof of a Program: Find*, in "Science of Computer Programming", To appear, 2001, http://www.lri.fr/~filliatr/ftp/publis/find.ps.gz.

[29] 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.

[30] 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.

[31] J.-C. FILLIÂTRE, F. POTTIER. *Producing All Ideals of a Forest, Functionally*, in "Journal of Functional Programming", vol. 13, n° 5, September 2003, p. 945–956, http://www.lri.fr/~filliatr/ftp/publis/kr-fp.ps.gz.

[32] 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.

[33] 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.

[34] 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/.

[35] N. OURY. *Observational Equivalence and Program Extraction in the Coq Proof Assistant*, in "TLCA", M. HOFMANN (editor)., Lecture Notes in Computer Science, vol. 2701, Springer, 2003, p. 271-285.

[36] 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.

[37] 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.

[38] X. URBAIN. *Modular and Incremental Automated Termination Proofs*, in "Journal of Automated Reasoning", 2004, http://www.lri.fr/~urbain/textes/jar.ps.gz.