INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team LEMME

Software and mathematics

Sophia Antipolis

THEME 2A

Table of contents

1. Team

Head of project-team
Loïc Pottier [Research scientist INRIA]

Vice-head of project team
Yves Bertot [Research scientist INRIA]

Administrative Assistant
Nathalie Bellesso

Staff members INRIA
Gilles Barthe [Research scientist INRIA]
Janet Bertot [Research engineer INRIA, service DREAM, at 60 %, until May 2003]
Marieke Huisman [Research scientist INRIA]
Laurence Rideau [Research scientist INRIA]
Laurent Théry [Research scientist INRIA (secondment University of L'Aquila, Italy)]

Civil servant (on partial secondment)
Philippe Audebaud [Lecturer, ENS Lyon]

Civil servant (on secondment)
Frédérique Guilhot [Qualified teacher, *académie de Nice*]

Post-doctoral fellows
Venanzio Capretta [Marie Curie fellowship, until October 2003]
Jan Cederquist [ERCIM fellowship, since March 2003, until December 2003]
Pierre Courtieu [Verificard, until September 2003]
Benjamin Grégoire [Concert, since November 2003]
Hanane Naciri [Mowgli]
Leonor Prensa [Profundis, until September 2003]
Igor Siveroni [since November 2003]
Christoph Sprenger [ERCIM fellowship/Modocop, until October 2003]

Scientific advisors
André Hirschowitz [Professor UNSA, Laboratoire J.A. Dieudonné]
Jean-Louis Lanet [INRIA DirDRI]
Roger Marlin [Professor UNSA, Laboratoire J.A. Dieudonné]
Monica Nesi [Lecturer, University of L'Aquila, Italy]

Software development staff
Lilian Burdy [ODL, since September 2003]

Ph.D. students
Antonia Balaa [Teaching Assistant UNSA, until September 2003]
Néstor Cataño [Verificard]
Laurent Chicli [Teaching Assistant UNSA]
Kuntal Das Barman [Verificard]
Guillaume Dufay [MESR grant, Teaching Assistant UNSA since September 2003]
Nicolas Magaud [MESR grant, until October 2003]
Assia Mahboubi [Teaching Assistant, since September 2003]
Milad Niqui [visiting PhD student, U. Nijmegen, March - May 2003]
Tamara Rezk [since February 2003]

Student interns
Lionel Alberti [ENS Cachan, 2 months]
Amitab Basu [IIT Delhi, Internships program, 2.5 months]

Jérôme Creci [DEA Paris 7, 6 months]
Damien Galliot [DEA Nice, 4 months]
Mariela Pavlova [DEA Paris 7, 6 months, Ph.D. student since December 2003]
Ando Saabas [Estonian grant, 4 months]
Sabrina Tarento [DEA Nice, 6 months, Ph.D student since October 2003]

2. Overall Objectives

Formal methods have become increasingly important in software development. The Lemme project aims at contributing to their use in software construction and scientific computing. In particular, we try to bridge the gap between solving a mathematical problem on paper and using a computer, as the latter requires many mechanical computations. Special attention is given to logical consistency and the ease of transfer from theory to practice.

To reach our goal, we work on the following themes:

1. development of the user's working environment in which the certified algorithms and the corresponding proofs are developed. Users can be, for example, researchers, engineers, or students;
2. formalization of mathematical theories describing the basic objects of scientific knowledge;
3. development of tools to facilitate the construction of mathematical proofs and efficient implementations, derived from the formal description of an algorithm and its correctness proof; and
4. formalization of programming language semantics, with special attention to the specification and verification of Java programs.

Although these themes could have a life of their own, they strongly influence each other. The researcher's working environment needs efficient and correct proof tools. Developing these tools requires the certified description of algorithms, which in turn requires formalizing the basic mathematical tools. Once algorithms are described, it is necessary to implement them in a programming language, whose semantics must be mastered. To complete the circle, formalizing mathematics or programming language semantics can be done more easily and efficiently thanks to a practical working environment.

In 2003, Gilles Barthe and Marieke Huisman started the process to create a new INRIA project: Everest (Environments for Verification and Security of Software). It is expected that this project will officially be created in 2004. Most of the research on programming language semantics will be continued within the Everest project.

3. Scientific Foundations

3.1. Proof environments

Key words: *proofs*, *environments*, *man-machine interface*, *Coq*.

We study how to improve mechanical tools for searching and verifying mathematical proofs used by engineers and mathematicians to develop software and formal mathematical theories. There are two complementary objectives. The first is to improve the means of interaction between users and computers, so that the tools become usable by engineers, who have otherwise little interest in proof theory, and by mathematicians, who have little interest in programming or other kinds of formal constraints. The second objective is to make it easier to maintain large formal mathematical developments, so they can be re-used in a wide variety of contexts. Thus, we hope to increase the use of formal methods in software development, both by making it easier for beginners and by making it more efficient for expert users.

3.2. Type theory and formalization of mathematics

Key words: *formalization*, *mathematics*, *type theory*, *Coq*.

The calculus of inductive constructions is powerful enough to formalize complex mathematics, based on algebraic structures together with their dependences and operations (as was also done in the Axiom computer algebra system). This is especially important as we want to produce proofs of logical properties for these structures, a goal that is only marginally addressed in most scientific computation systems. The calculus of inductive constructions also makes it possible to write algorithms as recursive functional programs, based on rich data structures. A third important characteristic of the calculus of inductive constructions is that it is also a language for manipulating proofs, thanks to the Curry-Howard isomorphism. All this makes the calculus of inductive constructions a tool of choice for our investigations. However, this language is difficult to learn, like an assembly language, although its conciseness and expressive power make it palatable for experts.

The Coq system and the graphical tool PCoq allow to have nicely readable formalizations: record structures, coercions, PPML pretty-printing, and extensible parsing, for instance, drastically improve readability. But numerous problems are left unsolved: multiple inheritance is not provided; stacking coercions leads to complexity problems; dependent types are awkward to use during proof or object constructions; sub-typing or quotient constructions are lacking.

3.3. Verification of scientific algorithms

Key words: *algorithms*, *certification*, *Coq*.

To produce certified algorithms, we use the following approach: instead of attempting to prove properties of an existing program (through a formalization of its semantics), we produce programs whose correctness is an immediate consequence of their construction. This has several advantages. First, we work at a high level of abstraction, independently of the target implementation language (but the closer the language to the functional approach, the more efficient the program). Second, we can concentrate on specific characteristics of the algorithm, and abstract away from the rest (*e.g.* memory management or data implementation strategies).

However, this approach also presents a few difficulties. For instance, it is still difficult to prove properties of recursive algorithms where termination is ensured thanks to a well-founded order. It is also difficult to work in an imperative style or with operations that have side-effects.

3.4. Programming language semantics

Key words: *semantics*, *programming languages*, *Java*, *Java Card*, *Coq*.

We also investigate the algorithms that occur when implementing programming languages. For these algorithms, we generally base our work on the semantic description of a language. The properties that we attempt to prove for an algorithm are, for example, that it preserves the semantics of programs (when the algorithm is a transformation or optimization algorithm) or that the programs produced are free of some unwanted behavior (when the algorithm is a compiler or a program verifier). For these algorithms, the complexity sometimes lies in the size of the language description, and sometimes in the intrinsic complexity of the algorithm, which may use other algorithms such as graph traversal or unification algorithms. We usually talk about "proofs in programming language semantics" to refer to this class of algorithms and their correctness proofs. In addition to its intrinsic interest, this work is also useful for our work on scientific algorithms. Using the semantics of the programming language used to implement the algorithms, we can guarantee the correctness and efficiency of these implementations.

4. Application Domains

4.1. Smart cards and other smart devices

The new generation of smart cards and other small personal devices typically contain a microprocessor and a memory chip (but with limited computing and storage capabilities). They are often used to store sensitive data, so reliability is an important issue for smart cards. This makes it a relevant application domain for the use of formal methods. To program such devices, high-level programming languages are used. In particular,

several dialects of Java exist for this purpose (Java Card, MIDP, *etc.*). In order to reason about the correctness of applications, both the platform and the programming language itself should be formalized.

Within the Lemme project, we have studied correctness proofs for byte code verifiers, focusing on object initialization and type safety. The current work is to develop appropriate tools to facilitate these correctness proofs. We also have carried out case studies to understand how to establish the correctness of typical smart device applications. This work is done in collaboration with the smart card producer Gemplus and within the context of the European IST project Verificard.

4.2. Certified scientific algorithms

For some applications, it is mandatory to build zero-default software. One way to reach this high level of reliability is to develop not only the program, but also a formal proof of its correctness. In the Lemme team, we are interested in certifying algorithms for scientific computing. For this, we propose a methodology that consists in starting not with the program but with the abstract algorithm. Proving first the algorithm's correctness has the main advantage that the proof usually contains all the deep mathematical properties. The second step that consists in deriving an efficient implementation from an algorithm could then be handled automatically or semi-automatically. We have already concluded several experiments in the area of computer arithmetic, polynomial computations, and computational geometry.

4.3. Web, MathML, XML

Our work around XML and MathML, within the context of Figue, has two important goals:

- it should enable us to share proofs (represented as Coq scripts) on the web, by generating XML + MathML representations of proofs from our PCoq interface. These XML+MathML proof representations can then be displayed (and printed) with "real" mathematical formulae by any navigator supporting MathML, thus making the proof representation independent of PCoq.
- continuing the work of Loïc Pottier on Wims (WWW Interactive Mathematics Server, developed by Xiao Gang at the University of Nice), we experiment with building proofs directly on the web.

In the long run, the shift of attention towards the web should increase the visibility of our work, so that we can have a larger group of users for our tools. The European contract LTR Mowgli will significantly support this.

5. Software

5.1. PCoq

Participants: Janet Bertot, Yves Bertot [correspondent], Loïc Pottier, Laurence Rideau.

We distribute the PCoq system, a front-end to the Coq theorem prover, which allows a better handling of mathematical notations. It has been used outside the team for several large scale proof developments, like the work on floating point numbers done at ENS Lyon. Work has been done to adapt PCoq and in particular the Figue component to produce and read/display XML/MathML text.

5.2. Aïoli and Figue

Participants: Hanane Naciri, Laurence Rideau [correspondent], Laurent Théry.

Aïoli and Figue are base components of PCoq. In PCoq, mathematical formulae are represented as trees. Aïoli defines the manipulation of these trees, while Figue defines appropriate algorithms for displaying. Particular attention is given to two-dimensional displaying of matrices, fractions, square roots, *etc.* For more information, see http://www-sop.inria.fr/lemme/aioli/doc/aioli.html and http://www-sop.inria.fr/lemme/figue/.

5.3. Tool for applet validation

Participants: Lilian Burdy [correspondent], Marieke Huisman.

We are developing a tool for applet validation. Its main design goals are:

- a high correctness assurance;
- an accessible user interface;
- prover independence; and
- a high degree of automation.

The tool takes Java programs, annotated with JML specifications. It generates appropriate proof obligations which can be used as input for different proof assistants (automatic or interactive).

The development of the tool is done in the context of an ODL (Software Development Support project) on generation of proof obligations and interactive proof systems.

5.4. Maximal model/applet generator

Participants: Dilian Gurov [KTH, Sweden], Marieke Huisman, Christoph Sprenger [correspondent].

This program derives a maximal transition system model from a given formula in a universal modal logic with greatest fixed points. The modal logic is adequate to express safety properties. The constructed model is maximal with respect to the simulation pre-order in the sense that it simulates all models satisfying the formula. In the context of applets, an interface is additionally given as input and the program constructs a maximal applet structure (a pushdown process specification) with the given interface and satisfying the given property. This program is used for the compositional verification of control flow based security properties of multi-applet Java Card application. It is written in Ocaml and is part of the Java Card Applet Verification Environment developed in collaboration with the Swedish Institute of Computer Science.

6. New Results

6.1. Tools for proof environments

6.1.1. PCoq

Participants: Philippe Audebaud, Janet Bertot, Yves Bertot, Frédérique Guilhot, Loïc Pottier, Laurence Rideau.

The PCoq system's evolution included adapting it to the most recent version of Coq (distributed during spring to make modules available), improvements on the history management, improvements on a tool to draw figures from geometrical statements (a description of this work was published in [13]), and preliminary experiments on an environment to help users tune the mathematical notations they use.

6.1.2. MathML rendering

Participants: Hanane Naciri, Laurence Rideau.

In the Mowgli context, we are interested in rendering structured data (proofs, mathematical papers) encoded in MathML. For this purpose, we have extended Figue, our interactive and two dimensional layout engine, in order to handle MathML data. Figue can be used by any Java application like a MathML rendering interface.

6.1.3. Latex to HTML converter

Participants: Hanane Naciri, Loïc Pottier.

We make scientific documents written in Latex accessible on the Web, while handling their structured objects like formulae. We first convert the Latex to XHTML+MathML, then convert it to HTML+images. For each formula, our program produces an image with specific sub-areas and associates each area with the

corresponding MathML structure (a sub formula) that can be manipulated. This gives mathematicians a simple way to obtain an active web version of their Latex documents, even if it is maybe not the most efficient way.

6.1.4. Proof explanations: using natural language and graph views

Participants: Frédérique Guilhot, Hanane Naciri, Loïc Pottier.

The aim of this work is to generate automatically from a Coq proof script (a set of commands given to Coq to perform the proof) an explanatory text in natural language (in French or in English) and a deduction graph. Hence formal proofs can be understood by people who are not familiar with proof assistants. To provide these views, we first translate the proof script into an XML tree. Then we visualize this tree as a structured text of explanations written using forward style (from assumptions to conclusion). These explanations are presented in a Web document with appropriate mathematical notations.

The XML tree can also be visualized using a deduction graph, an acyclic oriented graph in which each reasoning step is represented by a subgraph. In this view, only facts and the links between them are represented. This is a non linear presentation that gives a global view of the proof.

6.1.5. Mowgli prototype

Participants: Yves Bertot, Hanane Naciri, Laurence Rideau.

We have developed a first Mowgli prototype. The Mowgli prototype is a web interface giving access to the Mowgli library (which contains proof data from Coq and scientific papers in Tex source). This prototype gives proof developers the possibility to contribute to the Mowgli library. Proof developers can store their proofs in the Mowgli library, and browse through their own or other proofs. The Mowgli prototype gives both the content view (lambda terms, and proof trees in XML) and the natural language explanation view of the proof data (presented in HTML or XHTML+MathML). For more information, see http://www-sop.inria.fr/lemme/Hanane.Naciri/Mowgli/.

6.1.6. Classification of mathematical formulae

Participants: Hanane Naciri, Stephen Watt [ORCCA].

We continue our collaboration with ORCCA (Ontario Research Center for Computer Algebra) headed by Stephen Watt. We collaborate in particular on pen-based software for mathematics, recognizing hand-written mathematical formulae. We are building dictionaries for mathematical expressions for different mathematical fields (number theory, numerical analysis, group theory, ..). Each dictionary contains structured expressions and their frequencies. This makes it possible to determine the correct match among a list of ambiguous choices. We hope that this new experience will be beneficial to the treatment of mathematical documents within Mowgli.

6.1.7. Using Coq on the web

Participant: Loïc Pottier.

A *cgi* script has been implemented, allowing to use the Coq system directly from a standard browser. Its aim is to be used for teaching.

6.1.8. Geoview

Participants: Frédérique Guilhot, Loïc Pottier.

Geoview is a tool that allows to make a drawing from a theorem in planar geometry. It uses a Java applet (developed by F. Koteki) to show the drawing on the screen, and is integrated into the PCoq interface.

6.1.9. TeXMacs

Participants: Philippe Audebaud, Laurence Rideau.

Our environment for Coq using TeXmacs has been improved and adapted to the new version of Coq. It has been given the name TmCoq. Specific documentation of our project can be found at the address http://www-sop.inria.fr/lemme/Philippe.Audebaud/tmcoq/

From the user's point of view, we mainly provide a documentation tool `tmdoc` which translates any Coq script into an editable TeXmacs document. Therefore, it is also possible to browse between all the libraries and any user contribution from within the editor or (since very recently, owing to TeXmacs's exportation enhancements) from the Internet.

This work has been described in [8] and has been presented in the workshop *Math In Coq* organized by the Daemon Team in Orsay (April 23-24).

In addition, we have contributed to the Mowgli project by providing a generic Document Type Description (DTD) file for any TeXmacs document, and specific import/export plug-in with the XML format. This plug-in enables transparent external editing of our tmdoc-generated scripts, as well as of the Mowgli-related files. As such, this represents a first and successful experiment towards better integration of these projects.

6.2. Type theory and formalization of mathematics

6.2.1. Recursive functions

Participants: Antonia Balaa, Yves Bertot, Venanzio Capretta, Kuntal Das Barman, Nicolas Magaud.

The work of Antonia Balaa and Yves Bertot on recursive functions and their fix point function has been redesigned to give a more abstract treatment. We foresee that this new design will make it possible to adapt the technique to a wider class of recursive functions, including mutually recursive functions and nested recursive functions.

The technique of iterating functionals that is used in Antonia Balaa's work has been transposed to programming language semantics in a past work by Bertot, Capretta and Das Barman. This year, we have studied how this could be extended to obtain a tactic for proofs in programming language semantics using reflection. A paper on this topic has been submitted for publication [37].

A new class of recursive functions operating on co-inductive data has also been studied and we have experimented with solutions for the representation of these functions in the calculus of inductive constructions. This shows that the technique of ad-hoc recursion introduced by Ana Bove applies well in this domain, although general recursion and co-recursion need to be treated separately.

The work of Nicolas Magaud on the translation of proofs from one data structure to another has been extended in the direction of data structures based on dependent inductive types (like the type of fixed-length vectors). This work led to a publication [18] and the completion and defense of Nicolas Magaud's thesis [3].

6.2.2. Type theory

Participants: Yves Bertot, Pierre Castéran [U. Bordeaux/INRIA Futurs].

Yves Bertot and Pierre Castéran completed their book on the theoretical and practical aspects of the Coq System [31]. This work involved translating the whole text into English and adapting the examples to syntax changes in the Coq system.

6.2.3. Formalization in Coq of high-school geometry

Participants: Frédérique Guilhot, Loïc Pottier.

Frédérique Guilhot has developed and improved a library dedicated to high-school geometry for Coq. This library contains the formalizations of:

- several classical theorems in planar and spatial Euclidean geometry, among them: Thales, Desargues, Pythagoras, Simson's line, Miquel, Euler's line, nine-point circle;
- trigonometry;
- several properties of plane transformations: homothety, translation, reflection, rotation, direct similarity and inversion;
- analytic geometry; and
- complex numbers.

This work is reported in Research Report RR.4893 [25] and in an accepted article in JFLA 2004 [16].

6.2.4. Type-based termination of recursive definitions in higher-order and dependent settings

Participants: Gilles Barthe, Venanzio Capretta, Tarmo Uustalu [University of Tallinn].

G. Barthe, V. Capretta, and T. Uustalu have worked on scaling up type-based termination to higher-order and dependent settings, i.e., proving strong normalization of stage-annotated well-typed terms for appropriate higher-order, dependently typed lambda-calculi. We also made progress in automatic stage annotation inference for terms well-typed in a non-stage-annotated calculus without termination guarantees. This is meant to give a method to decide in a type-based fashion whether a general-recursive function definition is guarded-by-destructors recursive.

6.2.5. Recursive and non-well-founded coalgebras

Participants: Venanzio Capretta, Tarmo Uustalu [University of Tallinn], Varmo Vene [University of Tartu].

The concept of a recursive coalgebra (essentially a type equipped with a parsing operation and a recursion principle for definition of total functions) is due to Osius. Taylor's concept of a well-founded coalgebra captures admissibility of a closely related induction principle. Taylor has investigated the relationship of the two concepts to some extent, but foremostly for the category of sets. We identified sufficient conditions under which a recursive coalgebra is well-founded and vice versa in the general case. These results are relevant for totality analysis of general-recursive definitions in semantic universes other than Set; the proofs involve partial map theory. We also found a useful construction for manufacturing new recursive coalgebras from a given recursive coalgebra based on distributive laws.

6.2.6. Proving polynomial inequalities with real coefficients in Coq

Participants: Assia Mahboubi, Loïc Pottier.

We have implemented in Coq 7.4 a tactic, called Tarski, which proves polynomial inequalities with real coefficients, following an algorithm of Hörmander.

6.2.7. Proving polynomial equalities in Coq

Participants: Jérôme Creci, Loïc Pottier.

We have implemented in Coq 7.4 a tactic, called Gb, which proves polynomial equalities with coefficients in a domain. The tactic uses the extracted Ocaml code of the Coq implementation of Buchberger's algorithm, as developed by L. Théry in 1999.

6.3. Verification of scientific algorithms

6.3.1. Canonical representation of rational numbers and efficient algorithms

Participants: Yves Bertot, Milad Niqui.

In experiments from previous years, we devised a simple canonical representation of the ordered field of rational numbers (this was published in [12]). Milad Niqui, on visit from the University of Nijmegen (Netherlands), completed the formal description of a complex algorithm to compute efficiently the field operations. This proof required many of the tools and technique devised in the Lemme team, for instance Pierre Courtieu and Gilles Barthe's "Functional Inversion" procedure (included in the new release of Coq), and a new technique of "recursion on a ad-hoc predicate" as describe in Bertot and Castéran's book. In the long run, we hope that this work can be used to provide a formally proved library for exact real arithmetics.

A paper describing this work has been submitted for publication [26].

6.3.2. Graph coloring algorithms

Participants: Yves Bertot, Lionel Alberti.

We designed a formal description of a graph-coloring algorithm and proved that this algorithm is correct. This proof should be integrated in the certified compiler effort for the Concert New Investigation grant (ARC).

6.3.3. Verification of cryptographic algorithms

Participants: Jan Cederquist, Sabrina Tarento.

The aim of this work is to verify correctness of cryptographic algorithms. The formalization is based on the generic model (GM) and the random oracle model (ROM). The GM is characterized by the assumption that no specific structural properties of the underlying group representation can be exploited. These strong conditions allow a formalization without details about the representation of the group elements. The basic idea behind the ROM is that hash functions are chosen at random. Besides a formalization of the GM and the ROM, the main result is a proof that gives an upper bound to the probability of a non-interactive adversary (that adheres to the GM) to find a certain secret. The proof checker Coq has been used during the formalization and verification process.

6.4. Programming language semantics

6.4.1. Formal description of C- -

Participants: Yves Bertot, Kuntal Das Barman, Benjamin Grégoire, Laurence Rideau, Sandrine Blazy [IIE-CNAM].

The C- - programming language is a restricted fragment of the C programming language, which we designed as a good example of a non-trivial programming language that can be used for embedded software or for intermediate code in high-level programming language compilers.

Using an Ocaml specification of the C- - language (written by X. Leroy) as a guideline, we have formalized the language and its semantics in Coq. The semantics has been specified both through inductive definitions and through functions. Then we have proved that both specifications are equivalent. (Both specifications should be useful in the future for a proof of compiler correctness). The equivalence proof was an interesting challenge because the evaluator consists in five mutual recursive functions, and therefore a standard proof by induction is not possible. Therefore, we have used a method described by Bertot *et al.* in TPHOL'2002 Type-Theoretic Functional Semantics.

6.4.2. CertiCartes and Jakarta

Participants: Gilles Barthe, Pierre Courtieu, Guillaume Dufay, Simão Melo de Sousa.

We have extended our previous model of byte code verification to give formal executable descriptions of various byte code verifiers: a simple byte code verifier that handles monomorphic subroutines, a polymorphic byte code verifier that handles polymorphic subroutines, and a parameterized byte code verifier that generalizes both. Further, we have shown that all these BCVs reject programs that raise typing errors for the defensive JCVM.

6.4.3. Type systems for security

Participants: Gilles Barthe, Amitab Basu, Leonor Prensa, Tamara Rezk.

In the area of confidentiality, a type system for a large fragment of the JCVM has been produced, and shown to preserve termination-insensitive non-interference [30]. This is an important step in the understanding of non-interference for low-level languages, for which relatively little was known to date. An important issue is to ensure that security type systems for low-level languages relate appropriately to their counterparts for high-level languages. We have proven that compilation preserves security typing for a fragment of the Java Card source language [9]. To the best of our knowledge, this is the first such result in the literature, and the proof can be viewed as a procedure to compute, from a certificate of well-typing of the source program, another certificate of well-typing for the compiled program.

Using the proof assistant Isabelle/HOL, we have machine-checked a recent work of Boudol and Castellani, which defines an information flow type system for a concurrent language with scheduling, and shows that typable programs are non-interferent. As a benefit of using a proof assistant, we are able to deal with a more general language than the one studied by Boudol and Castellani, and to solve a (minor) flaw in the original

proof. The development, reported in [29], constitutes to our best knowledge the first machine-checked account of non-interference for a concurrent language.

6.4.4. Development of a tool for applet validation

Participants: Gilles Barthe, Lilian Burdy, Marieke Huisman, Tamara Rezk, Ando Saabas.

We are working on the development of a tool for applet validation. From September, Lilian Burdy has started working as the main developer of the tool. Several topics have been addressed to improve the effectiveness of the tool.

- Correctness: we add new tests and proof obligations to increase the precision of the tool: we implemented type checking for JML expressions and we added proof obligation generation for loop termination and well-definedness of JML expressions.
- Prover independence: we integrated the Simplify prover in the tool to increase the ratio of automatically proven proof obligations.
- User interface: we completed the user interface in Eclipse by associating a perspective to the existing plug-in and by adding a view in this perspective to display metrics of proven proof obligations.

At a more theoretical level, we also developed a weakest precondition-calculus for JML at byte code level, which will be integrated into the tool.

6.4.5. Slicing event spaces

Participants: Néstor Cataño, Marieke Huisman.

The goal of this work is to develop efficient techniques for checking temporal properties of multi-threaded Java programs. Based on the notion of event spaces, Cenciarelli *et al.* have developed an operational semantics for multi-threaded Java. This operational semantics forms the basis for our tool JEvent that generates an event space for a Java program.

To enable efficient checking of properties on event spaces, we adapted slicing techniques to the context of event spaces. In particular, this forces us to take aliasing into account. Currently, we are formalizing the slicing algorithm in PVS and we prove that slicing is property preserving.

6.4.6. Combining symbolic execution and run-time monitoring

Participants: Néstor Cataño, Willem Visser [NASA Ames], Corina Pasareanu [NASA Ames].

During the summer of 2003, Néstor Cataño has been an intern at NASA Ames, in the Automated Software Engineering group. During this internship, we studied techniques to show under which conditions no monitoring is required for a certain code fragment. Specifically, we show how to determine whether program statements will change the state of some observer, by doing a symbolic execution of the code with the Java PathFinder model checker. We only considered safety properties that are monitored by offline observers given as a finite-state automata. For example, one might like to monitor that "the temperature (denoted by variable `temp`) never exceeds 100 degrees"; hence the value of `temp` will be emitted to a monitor whenever `temp` is updated and the monitor automaton will make a transition when `temp > 100` is true (assuming that the temperature is initially less than or equal to `100` degrees).

6.4.7. Enforcing high-level security properties

Participants: Gilles Barthe, Lilian Burdy, Marieke Huisman, Jean-Louis Lanet, Mariela Pavlova, Igor Siveroni.

In the context of the DEA project of Mariela Pavlova, we developed a method to translate high-level security properties into a behavioral interface specification language, notably the Java Modeling Language (JML).

The method proceeds in two phases. First we synthesize appropriate annotations and second we weave them throughout the application. In this way, security policies can be validated using the various existing tools for

JML. The method is general and applies to a large class of security properties [38]. It has been implemented as a front-end for our own applet validation tool.

To illustrate its applicability, we applied the method to several realistic examples of smart card applications. This allowed us to find violations against the documented security policies for some of these applications.

Currently, we study extensions of the method, where security properties can be expressed using security automata. Further, we develop a formal proof that the generated annotations sufficiently characterize the security property.

6.4.8. Compositional verification of applet interactions

Participants: Gennady Chuganov [SICS,Sweden], Dilian Gurov [KTH, Sweden], Marieke Huisman, Christoph Sprenger.

A compositional verification method has been developed for behavioral control-flow properties of applets, based on a maximal model construction along the lines of previous work by Grumberg and Long. The maximal model construction is developed first in a general setting, using a special logic called simulation logic, and then instantiated to applets [27]. The maximal model construction has been implemented and integrated into the JCAVE applet control flow analysis tool set, developed at SICS. This tool set combines the compositional verification techniques with a number of third party tools for applet decompilation (SOOT) and model checking. The resulting verification technique is completely push-button and offers a general facility to formulate and check properties regarding temporal behaviors of applet properties.

To show the feasibility of our approach, we applied it to the Gemplus PACAP electronic purse case study. We used our techniques to prove the absence of unwanted interactions between the different applets provided by the case study [17].

6.4.9. Decomposing verification of multi-threaded Java programs

Participants: Marieke Huisman, Kerry Trentelman [ANU, Australia].

We developed a technique to reduce the verification of a temporal property for a multi-threaded Java application to the verification of the temporal property for parts of the application. As a side-condition one needs to show that the other parts of the application do not affect the validity of the temporal property. Often this can be done with a simple syntactic check, showing that the relevant variables are not affected. Decomposition of the application is done on the basis of the different possible execution threads. Currently, the rules are formalized and proven correct using the proof assistant Isabelle/HOL.

7. Contracts and Grants with Industry

7.1. Mowgli

We participate in the European LTR project Mowgli. Other participants are the Universities of Bologna (Italy), Berlin (Germany), Nijmegen (Netherlands) and Eindhoven (Netherlands), the DFKI (Saarbrücken, Germany), the Max Planck Institute (Germany), and Trusted Logic. The goal of the project is to build on top of previous standards for the management and publishing of mathematical documents (MathML, OpenMath, OMDoc), and to integrate them with different XML technologies (XSLT, RDF, *etc.*).

7.2. Verificard

We participated in the European IST project Verificard (January 2001 - September 2003). Other partners are the Universities of Nijmegen (Netherlands), Munich (Germany), Kaiserslautern (Germany) , SICS (Stockholm, Sweden) and Schlumberger. The goal of the project is the development of tools for the specification and verification of the Java Card platform and its applications.

7.3. Inspired

We participate in the European IST project Inspired (December 2003 - December 2006). The partners in this projects are all major industrial actors in the domain of smart cards, INRIA (also the projects Metiss and POPS) and the Universities of Louvain (Belgium) and Twente (Netherlands). The goal of the project is to define the new generation of Trusted Personal Devices. The role of INRIA is to develop appropriate formal methods for those.

7.4. Castles

RNTL Castles (Development of Static Analyses and Test for Secure Embedded Software, accepted in 2003, but not started yet). Other participants are Lande (Rennes), AQL and Oberthur. More information is available via http://www-sop.inria.fr/everest/projects/castles/. Goal of the project is to define an environment to support the formal specification and verification of low-level execution platforms

8. Other Grants and Activities

8.1. International collaborations

- ORCCA - Ontario Research Center for Computer Algebra, Canada : collaboration around MathML (bi-directional displaying of mathematical formulae).
- PAI Parrot with the University of Tallinn in Estonia. Collaboration on (co-)inductive types, classical operators and modalities in type theory.
- SICS and KTH, Stockholm, Sweden : compositional verification of control-flow security properties for applets.
- University of Nijmegen, Netherlands and ANU, Canberra, Australia: specification languages for Java (extending of JML towards high-level security properties).
- Universities of Bologna (Italy) and Nijmegen (Netherlands), DFKI (Saarbrücken, Germany) and the Max Planck Institute for Gravitational Physics (Germany): mathematics on the Web.

8.2. National initiatives

- INRIA New Investigation grant (ARC) Modocop (Model checking Of Concurrent Object-oriented Programs, 2002 - 2003). Other participants are Lande (Rennes), Oasis (Sophia), Vasy (Grenoble), Vertecs (Rennes) and the Distributed and Complex Systems Research of Verimag. The TFC team (Besançon) also regularly attends meeting. Several meetings have been held during the year, and the project finished with the organization of a Modocop workshop. For more information, see http://www-sop.inria.fr/lemme/modocop/.
- INRIA New Investigation grant (ARC) Concert (*Compilateur Certifié*, Certified compiler). Other participants are Cristal (Rocquencourt), Miró (Sophia and Nancy), Oasis (Sophia), Mimosa (Sophia), and IIE-CNAM (Evry). Several meetings have been held during the year, and the project shares a CVS repository and a mailing-list to gather data around the development of a certified compiler for C- -. For more information, see http://www-sop.inria.fr/lemme/concert/.
- ACI Sécurité GECCOO (Generation of Certified Code for Object-Oriented Applications - Specifications, refinement, proof and error detection, 2003 - 2006). Other participants are TFC (Besançon), Cassis (Nancy), LogiCal (LRI/Futurs) and Vasco (IMAG, Grenoble). For more information, see http://geccoo.lri.fr/.
- ACI Sécurité SPOPS (Secure Operating Systems for Trusted Personal Devices, 2003 - 2006). Other participants are POPS (Lille) and SSIR (Supélec, Rennes). For more information, see http://www-sop.inria.fr/everest/projects/spops/.
- COLOR Acces (Certified Cryptographic Algorithms). Other participants are I3S and the Laboratoire Jean Dieudonné. For more information, see http://www-sop.inria.fr/everest/projects/acces/.

8.3. European initiatives

Lemme participates in the networks Types (type theory) and Appsem (Applied Semantics).

9. Dissemination

9.1. Conference and workshop attendance, travel

Philippe Audebaud participated in a Mowgli meeting and the Mathematical Knowledge conference in Bertorino, Italy. Further he participated in TPHOLs/UITP (Rome, Italy) and Types (Torino, Italy).

Gilles Barthe participated and presented his work in POPL'03 (New Orleans, USA), RDP'03 (Valencia, Spain), APPSEM'03 (Nottingham, UK), SSA'03 (Montevideo, Uruguay), Verisafe (Rennes) and a Dagstuhl meeting on Language-Based Security (Germany).

Yves Bertot presented collective research work at the workshop "Semantic Web" in Eindhoven, Netherlands and his own work at the workshop "Mathematics, Logic, and Computation", a satellite workshop of ICALP'03 in Nijmegen, Netherlands. He attended the conferences JFLA'03 (Chamrousse), and TYPE'03 (Torino, Italy), TPHOLs/UITP 2003 (Rome, Italy) and the Mowgli workshop in Sophia.

Lilian Burdy participated in the conference FME 2003 (Pisa, Italy).

Venanzio Capretta participated in TPHOLs2003 (Rome, Italy). He visited the universities of Tallinn and Tartu, Estonia, for 10 days (28 August - 7 September).

Néstor Cataño presented his work at VMCAI (New York, USA) and FMICS (Trondheim, Norway). He attended the Verisafe meeting in Munich, Germany. He was a summer intern at NASA Ames in the Automated Software Engineering group (July - October 2003).

Pierre Courtieu presented his work at the Verisafe meeting in Munich, Germany.

Guillaume Dufay attended the Verisafe meeting in Munich, Germany.

Frédérique Guilhot presented Geoview and the geometry library during a meeting on formal geometry in Lyon.

Marieke Huisman presented her work during Trusted Components workshop (Prato, Italy), Verisafe meeting (Munich, Germany), and the Modocop workshop (Grenoble). She also participated in the last Verisafe meeting (Rennes). Further, she visited ANU, Canberra, Australia for three weeks, and gave a presentation at the seminar of the university of Besançon.

Jean-Louis Lanet visited several research institutes in Japan, to discuss Ubiquitous Computing. He was invited to give talks about his work on the 3rd meeting on Security of Information and Cryptography, and at the Modocop workshop.

Nicolas Magaud presented his work during Types'03 (Torino, Italy) and TPHOLs'03 (Rome, Italy). He also gave a presentation at the seminar of ENS Lyon. He participated in JFLA 2003.

Hanane Naciri participated in a Mowgli meeting and the conference Mathematical Knowledge in February (Bertinoro, Italy). In September she organized a Mowgli meeting in Sophia. She visited the ORCCA team (Ontario, Canada) from 4 to 26 October.

Mariela Pavlova presented her work at the Verisafe meeting in Rennes.

Loïc Pottier presented Geoview and the geometry library during a meeting on formal geometry in Lyon and during UITP (Rome, Italy).

Tamara Rezk presented her work at a seminar on Language-Based Security in Dagstuhl (Germany) and during the Verisafe meeting in Rennes. She participated in the Marktoberdorf summer school (Germany).

Christoph Sprenger presented his work during FoSSaCS in Warsaw, Poland, IEEE/ACM Memocode at Le Mont Saint Michel, and during a Modocop meeting. He visited KTH and SICS (Stockholm, Sweden) for a week.

9.2. Leadership within scientific community

- Kuntal Das Barman and Tamara Rezk took care of the organization of the internal Lemme seminar (see http://www-sop.inria.fr/lemme/Tamara.Rezk/seminaire/).
- Gilles Barthe co-organized the International Winter School on Semantics and Applications (SSA '03).
- Gilles Barthe was a member of the program committee for FMICS'03.
- Yves Bertot was a member of the program committees for TPHOLS'03 and UITP'03.
- Marieke Huisman edited a special issue of Journal of Logic and Algebraic Programming on formal methods for smart cards, together with Thomas Jensen (Lande).
- Project members reviewed project proposals for RNRT Commission 4 (software).
- Project members reviewed papers for among others JLAP, JAR, JFP, MSCS, TCS, ESOP and Types.

9.3. Miscellaneous

- Yves Bertot is a member of the jury for the SPECIF thesis award.
- Yves Bertot is a member of the *Conseil National des Universités* (National University Council).
- Loïc Pottier is a member of the *Conseil National des Universités* (National University Council), 27th section.

9.4. Visiting scientists

We had several visiting scientists, many of whom gave a talk in the Lemme seminar (cf. http://www-sop.inria.fr/lemme/Tamara.Rezk/seminaire/).

In particular, we mention the following visitors.

- Milad Niqui, University of Nijmegen, March - May 2003.
- Quang-Huy Nguyen, Loria, 2003.
- Claudio Sacerdoti and Luca Padovani, University of Bologna, 15 - 20 September 2003.
- Florian Kammüller, University of Berlin, 22 and 23 September 2003.
- David Naumann, Stevens Institute of Technology, 24 - 28 November 2003.
- Julien Narboux, Project LogiCal, 11 and 12 December 2003.

9.5. Supervision of Ph.D. projects

- Gilles Barthe supervises the Ph.D. projects of Maria João Frade (U. Minho, Portugal), Simão Melo de Sousa (completed February 2003), Guillaume Dufay (completed December 2003), Tamara Rezk and Sabrina Tarento.
- Gilles Barthe and Jean-Louis Lanet co-supervise the Ph.D. project Mariela Pavlova.
- Yves Bertot supervises the Ph.D. projects of Nicolas Magaud (completed October 2003) and Kuntal Das Barman. He also supervised Milad Niqui during his stay.
- Marieke Huisman supervises the Ph.D. project of Nestor Cataño.
- Loïc Pottier supervises the Ph.D. project of Assia Mahboubi and together with André Hirschowitz he supervises the Ph.D. project of Laurent Chicli (completed November 2003).

9.6. Ph.D. committees

- Gilles Barthe was member of the jury for the theses of Gerwin Klein, Francisco Gutierrez and Benjamin Grégoire.
- Jean-Louis Lanet was member of the jury for the thesis of Simão Melo de Sousa.

9.7. Supervision of internships

- Yves Bertot supervised Lionel Alberti, ENS Cachan, 2 months, and Damien Galliot, DEA Nice, 6 months.
- Gilles Barthe supervised Amitab Basu, IIT, India, 2.5 months, Sabrina Tarento, DEA Nice, 6 months, and Ando Saabas, University of Estonia, 4 months.
- Gilles Barthe, Lilian Burdy, Marieke Huisman and Jean-Louis Lanet co-supervised Mariela Pavlova, DEA Paris 7, 6 months.
- Loïc Pottier supervised Jérôme Creci, DEA Paris 7, 6 months.

9.8. Teaching

Antonia Balaa *Réseaux* (Networks, 1st year, 144 hours) IUT Génie des Télécommunications et Réseaux Sophia Antipolis.

Gilles Barthe *Sémantique et Sécurité* (Semantics and Security), DEA Info Nice.
Méthodes formelles pour cartes à puce (Formal methods for smart cards), Ecole Jeunes Chercheurs 2003, Aussois.
Formal verification techniques for Java Card, Winter School on Semantics and Applications.

Yves Bertot *Sémantique des langages de programmation* (Programming language semantics), Maîtrise Informatique (4th year, 10 hours), University of Nice.
Programmation fonctionnelle et preuves (Proofs and Functional Programming), Maîtrise, (4th year, 32 hours), ENS Lyon.

Guillaume Dufay *Algorithmique et Informatique Théorique* (Algorithms and Theoretical Computer Science), DEUG MASS 2 (2nd year, exercise groups, 39 hours)
Programmation, C et outils (Programming, C and tools), Licence Informatique, (3rd year, 26 hours).

Marieke Huisman *Sémantique des langages de programmation* (Programming language semantics), Maîtrise Informatique, (4th year, 6 hours), University of Nice.

Nicolas Magaud *Logique* (Logic), Licence Informatique, (3rd year, exercise groups 12 hours), University of Nice.
Programmation applicative (Applicative programming), Licence Informatique (3rd year, exercise groups, 10 hours), University of Nice.
Programmation applicative (Applicative programming), DEUG MI, (2nd year, practical training, 17 hours), University of Nice.

Hanane Naciri Algorithmic and Programming, DEUG MI (1st year, exercise groups, 21 hours), University of Nice.
Imperative programming, DEUG mathematics-physics (1st year, exercise groups, 17 hours), University of Nice.

Loïc Pottier *Preuves formelles: théorie et applications* (Formal Proofs: theory and applications), DEA mathematics, University of Nice.
Logique et informatique (Logic and Computer Science), Maîtrise MIM, University of Nice.

10. Bibliography

Doctoral dissertations and "Habilitation" theses

[1] L. CHICLI. *Sur la formalisation des mathématiques dans le calcul des constructions inductives.* Ph. D. Thesis, Université de Nice-Sophia-Antipolis, November, 2003.

[2] G. DUFAY. *Vérification formelle de la plate-forme Java Card.* Ph. D. Thesis, Université de Nice-Sophia-Antipolis, December, 2003.

[3] N. MAGAUD. *Changements de Représentation des Données dans le Calcul des Constructions.* Ph. D. Thesis, Université de Nice-Sophia-Antipolis, October, 2003.

[4] S. MELO DE SOUSA. *Outils et techniques pour la vérification formelle de la plate-forme JavaCard.* Ph. D. Thesis, Université de Nice-Sophia-Antipolis, February, 2003.

[5] L. POTTIER. *Quelques expériences de calculs et de raisonnements à l'aide de machines.* Thèse d'Habilitation à Diriger des Recherches, Université de Nice-Sophia Antipolis, 2003.

Articles in referred journals and book chapters

[6] G. BARTHE, V. CAPRETTA, O. PONS. *Setoids in Type Theory.* in « Journal of Functional Programming », volume 13, March, 2003, pages 261–293.

[7] C. SPRENGER, M. DAM. *A Note on Gobal Induction Mechanisms in a μ-Calculus with Explicit Approximations.* in « RAIRO - Theoretical Informatics and Applications », 2003, Accepted for publication (full version of FICS '02 paper).

Publications in Conferences and Workshops

[8] P. AUDEBAUD, L. RIDEAU. *TeXmacs as Authoring Tool for Publication and Dissemination of Formal Developments.* in « Proceedings of User Interfaces for Theorem Provers (UITP'03) », pages 14-32, 2003, Technical Report Nr.189, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, ISBN 88-7999-547-2.

[9] G. BARTHE, A. BASU, T. REZK. *Security Types Preserving Compilation.* in « Proceedings of VMCAI'04 », series Lecture Notes in Computer Science, volume 2xxx, Springer-Verlag, G. LEVI, B. STEFFEN, editors, 2004, to appear.

[10] G. BARTHE, H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Pure Pattern Type Systems.* in « Proceedings of POPL'03 », ACM Press, 2003.

[11] G. BARTHE, S. STRATULAT. *Using Implicit Induction Techniques for the Validation of the JavaCard Platform.* in « Proceedings of RTA'03 », series Lecture Notes in Computer Science, volume 2706, Springer-Verlag, R. NIEUWENHUIS, editor, pages 337 - 351, 2003.

[12] Y. BERTOT. *Simple canonical representation of rational numbers.* in « Mathematics, Logic and Computation, MLC 2003 », series Electronic Notes in Theoretical Computer Science, volume 85, Elsevier, H. GEUVERS,

F. KAMAREDDINE, editors, 2003.

[13] Y. BERTOT, F. GUILHOT, L. POTTIER. *Visualizing Geometrical Statements with GeoView.* in « Proceedings of User Interfaces for Theorem Provers (UITP'03) », pages 33-45, 2003, Technical Report Nr.189, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, ISBN 88-7999-547-2.

[14] L. CHICLI, L. POTTIER, C. SIMPSON. *Mathematical quotients and quotient types in Coq.* in « TYPES 2002 », series Lecture Notes in Computer Science, number 2646, Springer-Verlag, pages 95 - 107, 2003.

[15] J. CRECI, L. POTTIER. *La tactique GB.* in « JFLA 2004 », 2004, to appear.

[16] F. GUILHOT. *Formalisation en Coq d'un cours de géométrie pour le lycée.* in « JFLA 2004 », 2004, to appear.

[17] M. HUISMAN, D. GUROV, C. SPRENGER, G. CHUGUNOV. *Checking Absence of Illicit Applet Interactions: a Case Study.* in « Fundamental Approaches to Software Engineering (FASE'04) », series Lecture Notes in Computer Science, Springer-Verlag, 2004, to appear.

[18] N. MAGAUD. *Changing Data Representation within the Coq system.* in « Theorem Proving in Higher Order Logics (TPHOLS'03) », series LNCS, number 2758, Springer-Verlag, September, 2003.

[19] T. REZK, J. BLANCO. *Using Separation Logic to go from Functional Specification to Pointer Imperative Algorithms.* in « Proceedings of WAIT'03 », 2003.

[20] C. SPRENGER, M. DAM. *On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the μ-Calculus.* in « Foundations of Software Science and Computational Structures (FoSSaCS 2003), Warsaw, Poland, April 7-11, 2003, Proceedings », series LNCS, volume 2620, Springer, A. GORDON, editor, pages 425-440, 2003.

[21] C. SPRENGER, K. WORYTKIEWICZ. *A Verification Methodology for infinite-state message passing systems.* in « ACM & IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2003), Mont-Saint-Michel, France », June 24–26, 2003.

[22] N. CATAÑO. *Slicing Event Spaces: Towards a Java Programs Checking Framework.* in « FMICS: Eighth International Workshop on Formal Methods for Industrial Critical Systems », series Electronic Notes in Theoretical Computer Science, volume 80, Elsevier, T. ARTS, W. FOKKINK, editors, Trondheim, Norway, June 5-7, 2003.

[23] N. CATAÑO, M. HUISMAN. *Chase: A Static Checker for JML's Assignable Clause.* in « VMCAI: Verification, Model Checking and Abstract Interpretation », series Lecture Notes in Computer Science, volume 2575, Springer, L. D. ZUCK, P. C. ATTIE, A. CORTESI, S. MUKHOPADHYAY, editors, pages 26-40, New York, NY, USA, January 9-11, 2003.

Internal Reports

[24] C. BREUNESSE, N. CATAÑO, M. HUISMAN, B. JACOBS. *Formal Methods for Smart Cards: an experience report.* Technical report, number NIII-R0316, NIII, 2003.

[25] F. GUILHOT. *Premiers pas vers un cours de géométrie en Coq pour le lycée.* Technical report, number RR-4893, INRIA, 2003, http://www.inria.fr/rrrt/rr-4893.html.

[26] M. NIQUI, Y. BERTOT. *Qarith: Coq Formalisation of Lazy Rational Arithmetic.* Technical report, number RR-5004, INRIA, 2003, http://www.inria.fr/rrrt/rr-5004.html.

[27] C. SPRENGER, D. GUROV, M. HUISMAN. *Simulation Logic, Applets and Compositional Verification.* Technical report, number RR-4890, INRIA, 2003, http://www.inria.fr/rrrt/rr-4890.html.

Miscellaneous

[28] G. BARTHE, G. DUFAY. *A Tool-Assisted Framework for Certified Bytecode Verification.* October, 2003, Manuscript.

[29] G. BARTHE, L. PRENSA NIETO. *Formally Verifying Information Flow Type Systems for Concurrent and Thread Systems.* October, 2003, Manuscript.

[30] G. BARTHE, T. REZK. *Secure Information Flow for the JVM.* October, 2003, Manuscript.

[31] Y. BERTOT, P. CASTÉRAN. *Le Coq'Art.* 2003, http://www-sop.inria.fr/lemme/Yves.Bertot/coqart.html, Book to appear.

[32] A. BOVE, V. CAPRETTA. *Modelling General Recursion.* 2003, Submitted to Mathematical Structures in Computer Science.

[33] A. BOVE, V. CAPRETTA. *Recursive Functions with Higher Order Domains.* 2003, Manuscript.

[34] A. BOVE, V. CAPRETTA. *Representing Functional Programs in Impredicative Type Theory.* 2003, Manuscript.

[35] V. CAPRETTA. *A polymorphic representation of Induction-Recursion.* 2003, Manuscript.

[36] N. CATAÑO, C. PASAREANU, W. VISSER. *Combining symbolic execution and model checking to reduce dynamic program analysis overhead.* 2003, Submitted to Runtime Verification (RV'04).

[37] K. DAS BARMAN, Y. BERTOT. *Proof by reflection in semantics.* 2003, submitted for post-conference proceedings of TYPES'03.

[38] M. PAVLOVA, G. BARTHE, L. BURDY, M. HUISMAN, J.-L. LANET. *Enforcing High-Level Security Properties For Applets.* October, 2003, Manuscript.