INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Moscova

Mobility, Security, Concurrency, Verification and Analysis

Paris - Rocquencourt

THEME COM

Activity Report

2008

Table of contents

1. Team

Research Scientist

Jean-Jacques Lévy [Senior Researcher (DR) Inria]
Luc Maranget [Research Associate (CR) Inria]
Ricardo Corin [Research Associate (CR) Inria, since 1/10/2007]
James Leifer [Research Associate (CR) Inria]
Francesco Zappa Nardelli [Research Associate (CR) Inria]

PhD Student

Jade Alglave [MESR grant, Paris 7]
Pierre-Malo Deniélou [AMN, Paris 7]
Nataliya Guts [INRIA-MSR grant, Paris 7]
Jérémy Planul [ENS-Lyon, Ecole Polytechnique, from 1/10/2008 following 3-month internship]

Administrative Assistant

Sylvie Loubressac [Assistant (IE) Inria]

2. Overall Objectives

2.1. Overall Objectives

The research in Moscova centers around the theory and practice of concurrent programming in the context of distributed and mobile systems. The ambitious long-term goal of this research is the programming of the web or, more generally, the programming of global computations on top of wide-area networks.

The scientific focus of the project-team is the design of programming languages and the analysis and conception of constructs for security. While there have been decades of work on concurrent programming, concurrent programming is still delicate. Moreover new problems arise from environments now present with the common use of the Internet, since distributed systems have become heavily extensible and reconfigurable.

The design of a good language for concurrency, distribution and mobility remains an open problem. On one hand, industrial languages such as Java and $C\#$ allow downloading of programs, but do not permit migrations of active programs. On the other hand, several prototype languages (Facile [30], Obliq, Nomadic Pict [28], Jocaml, etc) have been designed; experimental implementations have also been derived from formal models (Join-calculus, Ambients, Klaim, Acute, etc). None of these prototypes has yet the status of a real programming language.

A major obstacle to the wide deployment of these prototype languages is the security concerns raised by computing in open environments. The security research addressed in our project-team is targeted to programming languages. It is firstly concerned by type-safe marshaling for communicating data between different run-times of programming languages; it is also related to the definition of dynamic linking and rebinding in run-times; it deals with versioning of libraries in programming languages; it is finally connected to access control to libraries and the safe usage of functions in these libraries.

We are also interested by theoretical frameworks and the design of programming constructs for transaction-based systems, which are relevant in a distributed environment. A theory of reversible process calculus has been studied in that direction.

On the software side, we pursue the development of Jocaml with additional constructs for object-oriented programming. Although the development of Jocaml is rather slow, due to the departure of several implementers and to interests in other topics, Jocaml remains one of our main objective in the next years.

The Acute [29] prototype, developed jointly at U. of Cambridge and at INRIA, has demonstrated the feasibility of our ideas in type-safe marshaling, dynamic binding and versioning; it is based on Fresh Ocaml, and the integration of these ideas in/with Jocaml will be also studied in the next years. Several other prototypes also appeared in 2007: OTT – one tool for the working semanticist, Burfiks – a Bayesian web filter. We also maintain Hevea, Advix, and the pattern-matching part of Ocaml.

In 2008, Gilles Peskine (presently at Trusted Logic) defended his PhD. Thomas Braibant (ENS-Lyon and MPRI) was intern with F. Zappa Nardelli. Jérémy Planul (ENS-Lyon and MPRI) was intern with Ricardo Corin and starts a PhD since september. Jade Alglave and Nataliya Guts finished their first year of PhD programme in Moscova.

Since August 2006, J.-J. Lévy is also director of the Microsoft Research-INRIA Joint Centre, in Orsay. J. Leifer, P.-M. Deniélou and F. Zappa Nardelli are also active in this centre, as members of the *Secure Distributed Computations and their Proofs*, headed by C. Fournet (Many members of the Joint Centre are former members of project-team Moscova).

Finally, we pursue the project PARSEC, funded by the ANR (*Agence Nationale de la Recherche*), together with MIMOSA, EVEREST, LANDE project-teams of INRIA and the team of Roberto Amadio at CNRS-PPS, U. of Paris 7. This project is coordinated by G. Boudol.

3. Scientific Foundations

3.1. Concurrency theory

Milner started the theory of concurrency in 1980 at Edinburgh. He proposed the calculus of communicating systems (CCS) as an algebra modeling interaction [25]. This theory was amongst the most important to present a compositional process language. Furthermore, it included a novel definition of operational equivalence, which has been the source of many articles, most of them quite subtle. In 1989, R. Milner, J. Parrow and D. Walker [26] introduced a new calculus, the *pi-calculus*, capable of handling reconfigurable systems. This theory has been refined by D. Sangiorgi (Edinburgh/INRIA Sophia/Bologna) and others. Many variants of the pi-calculus have been developed since 1989.

We developed a variant, called the Join-calculus [4], [5], a variant easier to implement in a distributed environment. Its purpose is to avoid the use of atomic broadcast to implement fair scheduling of processes. The Join-calculus allows concurrent and distributed programming, and simple communication between remote processes. It was designed with locations of processes and channels. It leads smoothly to the design and implementation of high-level languages which take into account low-level features such as the locations of objects.

The Join-calculus has higher-order channels as in the pi-calculus; channels names can be passed as values. However there are several restrictions: a channel name passed as argument cannot become a receiver; a receiver is permanent and has a single location, which allows one to identify channel names with their receivers. The loss of expressibility induced by these restrictions is compensated by joined receivers. A guard may wait on several receivers before triggering a new action. This is the way to achieve rendez-vous between processes. In fact, the notation of the Join-calculus is very near the natural description of distributed algorithms.

The second important feature of the Join-calculus is the concept of location. A location is a set of channels co-residing at the same place. The unit of migration is the location. Locations are structured as trees. When a location migrates, all of its sub-locations move too.

The Join-calculus, renamed Jocaml, has been fully integrated into Ocaml. Locations and channels are new features; they may be manipulated by or can handle any Ocaml values. Unfortunately the newer versions of Ocaml do not support them. We are still planning for both systems to converge.

3.2. Type systems

Types [27] are used in the theory of programming languages to guarantee (usually static) integrity of computations. Types are also used for static analysis of programs. The theory of types is used in Moscova to ensure safety properties about abstract values exchanged by two run-time environments; to define inheritance on concurrent objects in current extensions of Jocaml; to guarantee access control policies in Java- or $C\#$-like libraries.

3.3. Formal security

Formal properties for security in distributed systems started in the 90's with the BAN (Burrows, Abadi, Needham) logic paper. It became since a very active theory dealing with usual properties such as privacy, integrity, authentication, anonymity, repudiation, deniability, etc. This theory, which is not far from Concurrency theory, is relevant with the new activity of Moscova in the Microsoft Research-INRIA Joint Centre.

4. Application Domains

4.1. Telecoms and Interfaces

Keywords: *distributed applications*, *security*, *telecommunications*, *verification*.

Distributed programming with mobility appears in the programming of the web and in autonomous mobile systems. It can be used for customization of user interfaces and for communications between several clients. Telecommunications is an other example application, with active networks, hot reconfigurations, and intelligent systems. For instance, France Telecom (Lannion) designs a system programmed in mobile Erlang.

5. Software

5.1. OTT: One true tool for the working semanticist

Participants: Peter Sewell [U. of Cambridge], Francesco Zappa Nardelli.

Ott is a tool for writing definitions of programming languages and calculi. It takes as input a definition of a language syntax and semantics, in a concise and readable ASCII notation that is close to what one would write in informal mathematics. It generates as output:

- a LaTeX source file that defines commands to build a typeset version of the definition;
- a Coq version of the definition;
- an Isabelle version of the definition; and
- a HOL version of the definition.

Additionally, it can be run as a filter, taking a LaTeX/Coq/Isabelle/HOL source file with embedded (symbolic) terms of the defined language, parsing them and replacing them by typeset terms.

The main goal of the Ott tool is to support work on large programming language definitions, where the scale makes it hard to keep a definition internally consistent, and to keep a tight correspondence between a definition and implementations. We also wish to ease rapid prototyping work with smaller calculi, and to make it easier to exchange definitions and definition fragments between groups. The theorem-prover backends should enable a smooth transition between use of informal and formal mathematics.

In collaboration with Peter Sewell (Cambridge University).

The current version of Ott is about 24000 lines of OCaml. The tool is available from http://moscova.inria.fr/~zappa/software/ott (BSD licence).

Since its release in December 2007, the tool has been used in several projects, including a large proof of type preservation for the OCaml language (without modules) done by Scott Owens.

In 2008 we started the development of a new theorem-prover backend that will produce language definitions in "locally-nameless" style, thus supporting reasoning about alpha-equivalent terms. A paper describing the metalanguage used by Ott and its semantics has been accepted for publication in the Journal of Functional Programming [22].

5.2. Caml/Jocaml

Participant: Luc Maranget.

JoCaml is an implementation of the join-calculus integrated into Objective Caml (developed by team Gallium). With respect to previous join-language prototypes the most salient feature of the new prototype is a better integration into Objective Caml. We plan releases that will follow Objective Caml releases. See previous year repport for details on JoCaml. JoCaml is avaible at http://jocaml.inria.fr.

Last year (in June) saw the first public release of JoCaml. This year, as planned, we released a new version of JoCaml to match the new version of Objective Caml (version 3.11, december [21]). The new version also includes a few additions to JoCaml own library. More specifically, we provide two new modules that offer concurrent countdowns, and concurrents FIFOs.

Our general aim is now to promote JoCaml, by writing convincing examples, desribed in tutorials, and by providing extensive libraries to aid concurrent programming. Following the path initiated this year with the publication of a description of a concurrent ray tracer [16]. Our objective here is the promotion of a high-level style of programming, based upon higher order functions, parametric polymorphism and modules, all being inherited from Objective Caml, but being usable for programming concurrency. To that aim we intend to recruit manpower, ideally on a post-doctoral position.

5.3. Hevea

Keywords: *html*, *latex*, *tex*, *web*.

Participant: Luc Maranget.

Hevea is a translator from LaTeX to HTML, written in Objective Caml. Hevea was first released in 1997 and is still maintained and developed by Luc Maranget. A continuous (although informal) collaboration around Hevea exists, including Philip H. Viton (Ohio State University) for Windows port and Ralf Treinen (ENS Cachan) for Debian developments. For the record, Hevea consists in about 20000 lines of Caml and about 5000 lines of packages sources written in “almost TeX” (the language understood by Hevea).

This year saw a few developpements around hevea, mostly for maintenance.

5.4. Secure Sessions

Keywords: *functional languages*, *security*, *session*.

Participants: Karthik Bhargavan [Microsoft Research], Ricardo Corin, Cédric Fournet [Microsoft Research], James Leifer, Pierre-Malo Deniélou.

We have designed and implementation of a compiler that, given high-level multiparty session descriptions, generates custom cryptographic protocols. Our sessions specify pre-arranged patterns of message exchanges and data accesses between distributed participants. They provide each participant with strong security guarantees for all their messages.

Our compiler generates code for sending and receiving these messages, with cryptographic operations and checks, in order to enforce these guarantees against any adversary that may control both the network and some session participants. We furthermore verify that the generated code is secure by relying on a recent type system for cryptography. Most of the proof is performed by mechanized type checking, of the generated code, and does not rely on the correctness of our compiler. We obtain the strongest session security guarantees to date in a model that captures the actual details of protocol code.

Two central design goals guide our work on session implementation. First, all the cryptography required to protect compromised participants is completely hidden from the application programmer, who may reason about the behaviour of a distributed system as if it followed precisely the high level specification. (Thus all cor- respondence properties at the abstract level carry through to any distributed execution.) Second, all low-level network activity is in a one-to-one relationship with high-level communication, thus no additional messages are introduced.

Our compiler translates our session language to custom cryptographic protocols, coded as ML modules (both for $F\#$ with .NET cryptographic libraries, and for Ocaml with OpenSSL libraries), which can be linked to application code for each party of the protocol. Our compiler combines a variety of cryptographic techniques and primitives to produce compact message formats and fast processing.

The compiler consists of about 6000 lines of $F\#$. The trusted libraries for networking, cryptographic primitives, and principals shared by all session implementations have 780 lines of code (although their concrete implementation mostly relies on much-larger system libraries).

http://www.msr-inria.inria.fr/projects/sec/sessions/

5.5. Memevents-Litmus

Participant: Luc Maranget.

In november/december, Luc Maranget has entirely rewritten two tools which are instrumental to the "weak-memory" project, the model-checker (`memevents`) and the test environnements for actual machines (`litmus`), summing up about 4000 lines of Objective Caml.

In particular, he managed for these two tools (developped by different persons, with tight coupling) to share their front-end, resulting in the new possibility to run a specific test both against the model and on a real machine, and thus to automatize the consistency check of the memory models against the real machine (applicable for the x86 and Power architectures).

The tools are quickly evolving and no release is planned yet.

6. New Results

6.1. Join-calculus with values and pattern-matching

Participants: Luc Maranget, Ma Qin.

Last year, we wrote a journal version of previous work on mixing algebraic (à la ML) pattern matching and join-matching (published in the conference CONCUR in 2004).

In this work, we propose an extension of the join calculus with pattern matching on algebraic data types. Our initial motivation is twofold: to provide an intuitive semantics of the interaction between concurrency and pattern matching; to define a practical compilation scheme from extended join definitions into ordinary ones plus ML pattern matching. To assess the correctness of our compilation scheme, we develop a theory of the applied join calculus, a calculus with value passing and value matching. We implement this calculus as an extension of the current JoCaml system.

With respect to the previously published conference version, the extended version includes all proofs and a discussion of the implementation, which is of course based upon what we did for JoCaml.

The paper have been published in the Journal *Logical Methods in Computer Science* this year [13].

6.2. Concurrent C Minor

Participants: Andrew Appel [U. Princeton], Sandrine Blazy [ENSIIE and INRIA], Aquinas Hobor [University of Singapore], Francesco Zappa Nardelli.

Concurrent C Minor is an international research project to connect machine-verified source programs in sequential and concurrent programming languages to machine-verified optimising compilers. It takes inspiration from Xavier Leroy's design of the (sequential) C Minor language and verified compiler and has goals complementary to Leroy's CompCert project.

We designed a high-level intermediate language, Concurrent C Minor (CCm), with a small-step operational semantics representable in machine-checked proof assistants (we use Coq). We define a Concurrent Separation Logic for reasoning directly about source programs written in CCm. In addition, CCm is suitable as a target language for compilers from Java, C, $C\#$, ML, and other languages. This will allow safe and well-specified interaction of program modules written in different programming languages. We do not assume a sequentially consistent processor, and we formalise and reason about relaxed memory models.

Web page of the project: http://moscova.inria.fr/~zappa/projects/cminor.

In collaboration with Andrew Appel (Princeton University), Aquinas Hobor (University of Singapore), Sandrine Blazy (ENSIIE and INRIA Paris-Rocquencourt).

6.3. Secure Sessions

Participants: Karthikeyan Bhargavan[Microsoft Research-INRIA] „ Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet[Microsoft Research-INRIA] „ James Leifer, Jérémy Planul.

Distributed applications can be structured as parties that exchange messages according to some pre-arranged communication patterns. These sessions (or contracts, or protocols) simplify distributed programming: when coding a role for a given session, each party just has to follow the intended message flow, under the assumption that the other parties are also compliant.

In an adversarial setting, remote parties may not be trusted to play their role. Hence, defensive implementations also have to monitor one another, in order to detect any deviation from the assigned roles of a session. This task involves low-level coding below session abstractions, thus giving up most of their benefits.

We explore language-based support for sessions. We extend the ML language with session types that express flows of messages between roles, such that well-typed programs always play their roles. We compile session type declarations to cryptographic communication protocols that can shield programs from any low-level attempt by coalitions of remote peers to deviate from their roles. Our main result is that, when reasoning about programs that use our session implementation, one can safely assume that all session peers comply with their roles—without trusting their remote implementations.

Initial work was presented at CSF'07 [23], TGC'07 [24] and in a special issue of the Journal of Computer Security [12].

As extensions, we are working on two directions:

1. Support for integrity and secrecy support for a global store, and dynamic principal selection, which enables simple, abstract reasoning on global control and data flows. In this setting, we are developing novel, mostly-automated security proof techniques, where our compiler generates type annotations (from a predicate logic), which are then mechanically checked against actual executable code.

   A preliminary draft is available [18]. It has been submitted to IEEE S&P 2009.

2. Although sequential sessions are simple and intuitive, they lack flexibility and are in some cases too restrictive. Hence, for instance, concurrent, distributed programs must still be decomposed into series of smaller, sequential sessions, and the programmer is left to infer any security properties of the program from those guaranteed for its sub-sessions. In this line of work, we consider a more general class of session specifications that enable concurrency and synchronization *within a single session run*; and we systematically explore their secure implementation, thereby enforcing more global security properties in a distributed setting.

   Initial work constitued J. Planul's MPRI Master thesis at ENS Lyon [11], and further, preliminary work extending it (notably using a type system for checking implementability) is available soon.

6.4. Models of Audit Logs

Participants: Cédric Fournet [Microsoft Research], Nataliya Guts, Francesco Zappa Nardelli.

In an optimistic approach to security, one can often simplify protocol design by relying on audit logs, which can be analysed a posteriori. Such auditing is widely used in practice, but no formal studies guarantee that the log information suffices to reconstruct past runs of the protocol, to reliably detect, and provide evidence of, any cheating.

In a first work [15], we formalised audit logs for a sample optimistic scheme, the value commitment. It is specified in a pi calculus extended with committable locations, and compiled using standard cryptography to implement secure logs. We showed that our distributed implementation either respects the semantics of commitments or, using the information stored in the logs, detects cheating by a hostile environment.

Currently we study a general scheme to generate audit trails. Given an $F\#$ (a dialect of OCaml) program that implements some protocol, we discovered that the expected auditable properties can be specified using the F7 type system. We have thus defined a generic setup for auditability, and we have understood how type-checking can be used to check auditability. Preliminary investigations (described in [20]) suggest that from the type annotations it is possible to synthesise code that dumps the appropriate audit trail whenever the protocol is run.

6.5. Verified Implementations of Cryptographic Protocols

Participants: Karthik Bhargavan [Microsoft Research], Ricardo Corin, Cédric Fournet [Microsoft Research], Eugen Zalinescu [MSR-INRIA].

In this work carried in collaboration with C. Fournet, K. Bhargavan (MSR Cambridge) and E. Zalinescu (MSR-INRIA), we intend to narrow the gap between concrete implementations and verified models of cryptographic protocols. To this end, we are considering protocols implemented in ML and verified using CryptoVerif, Blanchet's protocol verifier for computational cryptography. We experiment with compilers from ML code to CryptoVerif processes, and from CryptoVerif declarations to ML code.

Preliminary work appeared at FCC07 [19].

We have used our compiler to verify the Transport Layer Security protocol (TLS). In that work [14], we programmed a small functional implementation of TLS that interoperates with mainstream implementations.

Relying on a combination of model-extraction and verification tools, we obtain a range of positive security results, covering both symbolic and computational cryptographic aspects of the protocol. We thus provide security guarantees for code as it is used in typical deployments of TLS.

We are currently working on extending that work. We show the correctness of our translation, with respect to a probabilistic, polynomial-time semantics for ML. This enables us to carry over the computational properties verified by CryptoVerif to our source programs, in terms of PPT adversaries with access to selected ML interfaces.

We also improve on libraries that rely on private databases to store local state and cryptographic materials for principals. This programming style is delicate to translate to CryptoVerif, which does not support private channels. We are considering models that combine local variable bindings (for data writes) and commands for data lookups.

More information, including the prototype compiler and supporting files for the examples, is available at the Project homepage (http://www.msr-inria.inria.fr/projects/sec/fs2cv/index.html).

6.6. The Semantics of Multiprocessor Machine Code

Participants: Jade Alglave, Thomas Braibant [MPRI, ENS-Lyon], Luc Maranget, Francesco Zappa Nardelli.

Multiprocessors are now dominant, but real multiprocessors do not provide the sequentially consistent memory that is assumed by most work on semantics and verification. Instead, they have subtle relaxed (or weak) memory models, usually described only in ambiguous prose, leading to widespread confusion.

We have developed rigourous and accurate semantics for multiprocessor programs above three architectures: x86, Power, and ARM. Each covers the relaxed memory model, instruction semantics, and instruction decoding, for fragments of the instruction sets, and is mechanised in HOL. This should provide a solid intuition for low-level programming, and a sound foundation for future work on verification, static analysis, and compilation of low-level concurrent code, and for the design of high-level concurrent programming languages.

For x86, we tested the semantics against actual processors and the vendor litmus-test examples, and gave an equivalent abstract-machine characterisation of our axiomatic memory model. We showed that the memory model is equivalent to a simpler ('nice execution') definition. For programs that are, in some precise sense, data-race free, we proved in HOL that their behaviour is sequentially consistent. The model is, to the best of our knowledge, consistent with the current Intel and AMD documentation: the Intel 64 and IA-32 Architectures Software Developer's Manual [vol 1,2A,2B,3A,3B, rev.28, Sept. 2008] and AMD64 Architecture Programmer's Manual [vol 2, rev. 3.14, Sept. 2007], and also with the earlier Intel SDM [vol 1,2A,2B, rev.27, April 2008; vol.3A,3B, rev.26, Feb. 2008] and Intel 64 Architecture Memory Ordering White Paper [Aug. 2007]. However, there are outstanding questions as to whether these are good descriptions of actual x86 processors: see the Addendum to the x86-CC paper below.

The Power and ARM architectures have a very different memory model to x86, but are very similar to each other. Our semantics is, to the best of our knowledge, consistent with the Power 2.05 specification and the ARM Architecture Reference Manual v7. Here the core memory model is also formalised in Coq.

Web page of the project: http://moscova.inria.fr/~zappa/projects/weakmemory.

In collaboration with Peter Sewell, Susmit Sarkar, Scott Owens, Tom Ridge, Magnus Myreen (Cambridge University), and Samin Isthiaq (Microsoft Research Cambridge).

6.7. ML pattern-matching

Participant: Luc Maranget.

Luc Maranget wrote a paper which provides a complete study of the technique of compiling ML-pattern matching to decision trees. The paper has been published at the occasion of the ML workshop affiliated to ICFP'08 [17].

The paper addresses the issue of compiling ML pattern matching to compact and efficient decisions trees. Traditionally, compilation to decision trees is optimized by (1) implementing decision trees as dags with maximal sharing; (2) guiding a simple compiler with heuristics. We first design new heuristics that are inspired by *necessity*, a concept from lazy pattern matching that we rephrase in terms of decision tree semantics. Thereby, we simplify previous semantic frameworks and demonstrate a straightforward connection between necessity and decision tree runtime efficiency. We complete our study by experiments, showing that optimizing compilation to decision trees is competitive with other optimizing match compilers, which are much more difficult to design and program.

As stated above, the interest of resurrecting the old technique of compiling to decision trees is the simplicity of the compiler itself. This may, for instance, lead to advances in the formal proof of a ML compiler that would nevertheless produce compact and efficient code.

7. Other Grants and Activities

7.1. National actions

7.1.1. PARSEC

We started at end of 2006 a new project PARSEC, funded by the ANR (*Agence Nationale de la Recherche*), together with MIMOSA, EVEREST, LANDE project-teams of INRIA and the team of Roberto Amadio at CNRS-PPS, U. of Paris 7. This project is coordinated by G. Boudol. This project is about the design of programming languages for distributed applications and their security properties.

7.2. European actions

7.2.1. Collaboration with Microsoft

In 2006, we started to work at the Microsoft Research-INRIA Joint Centre in a common project with Cédric Fournet (MSR Cambridge), Gilles Barthe, Benjamin Grégoire and Santiago Zanella (INRIA Sophia-Antipolis). The project is named *Secure Distributed Computations and their Proofs* and deals with security, programming languages theory and formal proofs. This work was still under active collaboration within all year 2008.

8. Dissemination

8.1. Animation of research

R. Corin was a member of the Program Committee for the IEEE ARES 2008 security conference.

J. Leifer was a member of the Program Committee for POPL (Principles of Programming Languages), which takes place in Savannah, Georgia, in January 2009 [170 submissions for 36 accepted papers]. This required several months of work. The PC members reviewed the submissions "blind" without reference to outside expert subreviewers before incorporating their comments in the review process.

J. Leifer proofread and edited chapters of INRIA's *Strategic Plan*, which was presented to the Government in 2008.

J.-J. Lévy is director of the Microsoft Research-INRIA Joint Centre, see http://www.msr-inria.inria.fr/. He participated to the start of the 4 new research teams of so-called track B (Computational Sciences). He co-organised the official opening of the Centre on Jan 11, 2007 and chaired 4 Management Committees with representatives of INRIA and Microsoft Research Cambridge.

L. Maranget is elected member of *Comité technique paritaire* of Inria, 1 meeting every 2 months about the general politics of Inria.

J. Leifer was on the thesis committee of Gilles Peskine whom he cosupervised with Jean-Jacques Lévy. Peskine successfully defended his thesis "Abstract types in collaborative programs" (l'Université Paris 7 (Denis Diderot)) on June 12, in front of a jury consisting of J.-J. Lévy, Robert Harper, Jacques Garrigue, J. Leifer, Didier Rémy, Peter Sewell, and Roberto Di Cosmo (president).

J.-J. Lévy is member of the Scientific Committee of the "Fondation Sciences Mathématiques de Paris" and participates to corresponding meetings.

J.-J. Lévy is member of the Program Committee of Digiteo as representative of INRIA–Paris-Rocquencourt, since September.

8.2. Teaching

Our project-team participates to the following courses:

- "Concurrency", Master Parisien de Recherche en informatique (MPRI), 2008-2009, at U. of Paris 7, 15 students, (F. Zappa Nardelli taught the pi-calculus semantics: 12 hours and the final examination);
- 'Weak memory models", F. Zappa Nardelli, IMDEA, Madrid, Spain (6 hours), September, 22-26;
- "Informatique Fondamentale 1 (IF1)", 1st year course at U. of Paris 7 Denis Diderot, 2 groups (23 + 27 students, 3 + 3 hours a week) among 370 students (P.-M. Deniélou did the lab classes on Java programming).
- "Introduction to Programming Languages Theory", Master 1 course at the Ecole polytechnique, 14 students, 9×1.5 hours for classes and 9×2 hours for laboratory classes, plus the final examination. L. Maranget is Professeur Chargé de Cours (part time) at École polytechnique. Teaching material is at http://www.enseignement.polytechnique.fr/profs/informatique/Luc.Maranget/

We also had the following activity related to teaching:

- L. Maranget coordinates the 3 computer science problems (4h+2h+2h) of the entrance examination at the Ecole polytechnique in 2009.

8.3. Participations to conferences, Seminars, Invitations

8.3.1. Participations to conferences

- January, 22-25, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- January 16, F. Zappa Nardelli gave a talk at ENS-Lyon.
- March, 30-April 2, F. Zappa Nardelli attended the ESOP conference, Budapest, Hungary.
- March, 30-April 2, J. Alglave attended the ESOP conference, Budapest, Hungary.
- March, 30-April 3, N. Guts attended the ESOP conference, Budapest, Hungary. She presented the work “A Formal Implementation of Value Commitment”.
- March, 29- Avril 6, L. Maranget presented his papers [16], [17] at the ESOP 2008 conference, Budapest.
- May, 13-15, J. Leifer attended the 3rd Annual XVR Workshop – Virtual Environments and Brain Computer Interfaces at the Scuola Superiore Sant’Anna, Pisa, Italy.
- June, 24, F. Zappa Nardelli attended the Master defence of Thomas Braibant at ENS-Lyon.
- June, 9, F. Zappa Nardelli attended the fourth meeting of the ANR ParSec Project at IRISA, Rennes. He gave a talk on “The Intel 64/ia-32 relaxed memory model”.
- June, 9, J. Vitek attended the fourth meeting of the ANR ParSec Project at IRISA, Rennes. He gave a talk on “Flexible Task Graphs”.
- July, 22-30, N. Guts attended the summer school on “Logic and Theorem Proving in Programming Languages”, Eugene, Oregon.
- August, 5-17, N. Guts attended the Marktoberdorf summer school on “Engineering Methods and Tools for Software Safety and Security”, Martkoberdorf, Germany.
- September 7-13, J. Alglave attended the LASER Summer School on Software Engineering, Italy.
- September 22-24, L. Maranget presented his work on Decision Trees for ML pattern-matching at the ML 2008 workshop, Victoria, British Columbia, Canada.
- September, 22-26, F. Zappa Nardelli visited IMDEA, Spain, for collaboration with Gilles Barthe. He lectured on “Weak memory models” and gave a talk on “the Ott tool”.
- September 22-29, J. Leifer attended ICFP 2008 (The 13th ACM SIGPLAN International Conference on Functional Programming) in Victoria, British Columbia, Canada, along with associated workshops, and the POPL PC meeting colocated at the conference centre.
- November 19-21, J. Leifer attended ENACTIVE08, the 5th International Conference on Enactive Inferfaces at the Scuola Superiore Sant’Anna, Pisa, Italy.
- December, 7-9, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.

8.3.2. Other Talks and Participations

- Jan 28, L. Maranget talked about JoCaml at the *Université de Créteil* (Jan. 28).
- Apr 19, J.-J. Lévy talked about Theory of Computation (Computability) at the Lycée Jules Verne in Cergy with an audience of students of “Terminales scientifiques et technologiques”.

- Jun 2-3, J.-J. Lévy participated to the TAB (Technical Advisory Board) meeting at Microsoft Research (MSR), Cambridge, and gave a presentation of current research at the MSR-INRIA Joint Centre.
- Sep 26, J.-J. Lévy gave a presentation about the Ariane 5 1996 debugging story at the Unithe in Saclay (monthly talk for INRIA – Saclay).
- Oct 13, L. Maranget gave a presentation on decision trees at the GALIUM-MOSCOVA seminar.
- Oct 22, J.-J. Lévy gave the same presentation for the mathematics professors of the Académie de Versailles.
- Nov 4, J.-J. Lévy was invited to give a talk about current research at the MSR-INRIA Joint Centre at the Pôle El Gazala des Technologies de la Communication, Tunis, Tunisia.
- Nov 18, P.-M. Deniélou visited the University of Southampton, UK for collaboration with Vladimiro Sassone.
- Nov 19, P.-M. Deniélou visited the Queen Mary University of London, UK and gave a talk on “Protocol Synthesis for Secure Sessions in ML”.
- Nov 20, P.-M. Deniélou visited the Imperial College for collaboration with Nobuko Yoshida.

8.3.3. Visits

- April and May, P.-M. Deniélou did an internship at the Microsoft Research Laboratory in Cambridge, UK, under the supervision of Cédric Fournet and Karthikeyan Bhargavan. The subject of the internship was “Protocol Synthesis for Secure Sessions”.
- May 15-July 15, Jan Vitek spent two months in the Moscova Project-Team for collaboration with F. Zappa Nardelli.
- Jun 10-12, Peter Sewell visited Rocquencourt for collaboration with F. Zappa Nardelli and J. Alglave.
- Jun 10-12, Susmit Sarkar visited Rocquencourt for collaboration with F. Zappa Nardelli and J. Alglave.
- Thomas Braibant did a M2 (master) internship during six months (February-July) under the supervision of F. Zappa Nardelli.
- Jérémy Planul did a M2 (master) internship during six months (February-July) in Moscova and the MSR-INRIA laboratory under the supervision of R. Corin.

9. Bibliography

Major publications by the team in recent years

[1] P. NING, P. F. SYVERSON, S. JHA (editors). *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, ACM, 2008.

[2] F. BESSON, T. BLANC, C. FOURNET, A. D. GORDON. *From Stack Inspction to Access Control: A Security Analysis for Libraries*, in "17th IEEE Computer Security Foundations Workshop", June 2004, p. 61–75.

[3] R. CORIN, P.-M. DENIÉLOU, C. FOURNET, K. BHARGAVAN, J. J. LEIFER. *Secure Implementations for Typed Session Abstractions*, in "20th IEEE Computer Security Foundations Symposium (CSF'07), Venice, Italy", IEEE, July 2007, p. 170–186, http://www.msr-inria.inria.fr/projects/sec/sessions/.

[4] C. FOURNET, G. GONTHIER. *The Reflexive Chemical Abstract Machine and the Join-Calculus*, in "Proceedings of the 23rd Annual Symposium on Principles of Programming Languages (POPL) (St. Petersburg Beach, Florida)", ACM, January 1996, p. 372–385.

[5] C. FOURNET, G. GONTHIER, J.-J. LÉVY, L. MARANGET, D. RÉMY. *A Calculus of Mobile Agents*, in "CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)", U. MONTANARI, V. SASSONE (editors), LNCS, vol. 1119, Springer, 1996, p. 406–421.

[6] C. FOURNET, C. LANEVE, L. MARANGET, D. RÉMY. *Inheritance in the join calculus*, in "Journal of Logics and Algebraic Programming", vol. 57, n^{o} 1–2, September 2003, p. 23–29.

[7] A. HOBOR, A. APPEL, F. ZAPPA NARDELLI. *Oracle Semantics for Concurrent Separation Logic*, in "17th European Symposium on Programming (ESOP'08)", April 2007.

[8] J. J. LEIFER, G. PESKINE, P. SEWELL, K. WANSBROUGH. *Global abstraction-safe marshalling with hash types*, in "Proc. 8th ICFP", Extended Abstract of INRIA Research Report 4851, 2003, http://hal.inria.fr/inria-00071732/fr/.

[9] M. MERRO, F. ZAPPA NARDELLI. *Behavioural theory for Mobile Ambients*, in "Journal of ACM", vol. 52, n^{o} 6, November 2005, p. 961–1023.

[10] M. QIN, L. MARANGET. *Expressive Synchronization Types for Inheritance in the Join Calculus*, in "Proceedings of APLAS'03, Beijing China", LNCS, Springer, November 2003.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] J. PLANUL. *Secure Local Enforcement for Global Process Specifications*, MPRI thesis, ENS Lyon, July 2008.

Articles in International Peer-Reviewed Journal

[12] R. CORIN, P.-M. DENIÉLOU, C. FOURNET, K. BHARGAVAN, J. J. LEIFER. *A secure compiler for session abstractions*, in "Journal of Computer Security", vol. 16, n^{o} 5, 2008, p. 573-636.

[13] M. QIN, L. MARANGET. *Algebraic Pattern Matching in Join Calculus*, in "Logical Methods in Computer Science", vol. 4, March 2008, 41, http://www.lmcs-online.org/.

International Peer-Reviewed Conference/Proceedings

[14] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Cryptographically verified implementations for TLS*, in "ACM Conference on Computer and Communications Security", 2008, p. 459-468.

[15] CÉDRIC. FOURNET, N. GUTS, F. ZAPPA NARDELLI. *A Formal Implementation of Value Commitment*, in "Proc. ESOP 2008", LNCS, vol. 4960, Springer-Verlag, 2008, p. 383–397.

[16] L. MANDEL, L. MARANGET. *Programming in JoCaml*, in "European Symposium on Programming (ESOP'08)", Tool presentation, LNCS 4960, April 2008.

[17] L. MARANGET. *Compiling Pattern Matching to Good Decision Trees*, in "Workshop on the language ML", ACM Press, 2008.

Other Publications

[18] *Cryptographic Protocol Synthesis and Verification for Multiparty Sessions*, http://www.msr-inria.inria.fr/projects/sec/sessions/cryptographic-protocol-synthesis-and-verification-for-multiparty-sessions.pdf.

[19] K. BHARGAVAN, C. FOURNET, R. CORIN, E. ZALINESCU. *Cryptographically verified implementations for TLS*, in "Formal and Computational Cryptography Workshop", 2008.

[20] CÉDRIC. FOURNET, N. GUTS, F. ZAPPA NARDELLI. *Auditability, Evidence, and Out-of-Context Proofs*, Draft, 2008.

[21] L. MARANGET, L. MANDEL, M. QIN. *JoCaml, release 3.11*, December 2008, http://jocaml.inria.fr/.

[22] P. SEWELL, F. ZAPPA NARDELLI, S. OWENS, G. PESKINE, T. RIDGE, S. SARKAR, R. STRNIŠA. *Ott: Effective Tool Support for the Working Semanticist*, Accepted for publication in the Journal of Functional Programming.

References in notes

[23] R. CORIN, P.-M. DENIÉLOU, C. FOURNET, K. BHARGAVAN, J. J. LEIFER. *Secure Implementations for Typed Session Abstractions*, in "20th IEEE Computer Security Foundations Symposium (CSF'07), Venice, Italy", IEEE, July 2007, p. 170–186, http://www.msr-inria.inria.fr/projects/sec/sessions/.

[24] R. CORIN, P.-M. DENIÉLOU. *A Protocol Compiler for Secure Sessions in ML*, in "TGC", G. BARTHE, C. FOURNET (editors), Lecture Notes in Computer Science, vol. 4912, Springer, 2007, p. 276-293.

[25] R. MILNER. *Communication and Concurrency*, International Series on Computer Science, Prentice Hall, 1989.

[26] R. MILNER, J. PARROW, D. WALKER. *A Calculus of Mobile Processes, Parts I and II*, in "Journal of Information and Computation", vol. 100, September 1992, p. 1–77.

[27] B. C. PIERCE. *Types and Programming Languages*, The MIT Press, 2002.

[28] B. C. PIERCE, D. N. TURNER. *Pict: User Manual*, Available electronically, 1997.

[29] P. SEWELL, J. J. LEIFER, K. WANSBROUGH, F. ZAPPA NARDELLI, M. ALLEN-WILLIAMS, P. HABOUZIT, V. VAFEIADIS. *Acute: High-level programming language design for distributed computation*, in "Journal of Functional Programming", vol. 17, n^{o} 4-5, 2007, p. 547–612.

[30] B. THOMSEN, L. LETH, T.-M. KUO. *A Facile Tutorial*, in "CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)", U. MONTANARI, V. SASSONE (editors), LNCS, vol. 1119, Springer, 1996, p. 278–298.