INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Moscova

Mobility, Security, Concurrency, Verification and Analysis

Paris - Rocquencourt

Theme : Programs, Verification and Proofs

Table of contents

1. Team

Research Scientist

Jean-Jacques Lévy [Team leader, Senior Researcher (DR) Inria]
Luc Maranget [Research Associate (CR) Inria]
Karthikeyan Bhargavan [Research Associate (CR), since 1/10/2009]
Ricardo Corin [Research Associate (CR) Inria, until 1/3/2009]
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, until 1/10/2009]
Nataliya Guts [INRIA-MSR grant, Paris 7]
Jérémy Planul [ENS-Lyon, Ecole Polytechnique]

Post-Doctoral Fellow

Giulio Manzonetto [Post-Doctoral Fellow, until 30/11/2009]

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 [29], 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.

OTT – one tool for the working semanticist – is still under development with a growing set of users. A prototype for secure multiparty sessions has been developped in collaboration with Microsoft Research (MSR) Cambridge. A suite of tools for analysis of weak memory models is also under construction in our group.

In 2009, Pierre-Malo Deniélou left for Imperial College, London (he will defend his PhD on 25 January 2010). Karthik Bhargavan (previously at MSR Cambridge) has now an INRIA CR1 position at Moscova. Jérémy Planul (ENS-Lyon and MPRI) accomplished his first year of PhD. Jade Alglave and Nataliya Guts starts their third year of PhD.

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

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 output:

1. a LaTeX source file that defines commands to build a typeset version of the definition;
2. a Coq version of the definition;
3. an Isabelle version of the definition; and
4. 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 2009 we completed the development of a new theorem-prover backend that produces language definitions in "locally-nameless" style, thus supporting reasoning about alpha-equivalent terms. A paper describing the metalanguage used by Ott and its semantics (including the new backend) will appear in the Journal of Functional Programming [19].

5.2. Caml/Jocaml

Participants: Xavier Clerc, 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 achieve binary compatibility with Objective Caml, and plan releases that will follow Objective Caml releases. See previous years reports for details on JoCaml. JoCaml is avaible at http://jocaml.inria.fr. Current version is 3.11.1 (released in June).

Xavier Clerc, INRIA engineer, now works part-time on JoCaml. The objective is to extend the JoCaml library to provide easier access to concurrency and distribution. In particular Xavier Clerc wrote:

1. a set of library fonctions to clone JoCaml program,
2. a package a la *map/reduce*.

These extensions will be included in the next release of JoCaml.

5.3. Secure Sessions

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

5.4. Memevents-Litmus-Diy

Participants: Jade Alglave, Luc Maranget, Susmit Sekar [U. of Cambridge].

The tools suite of the *Weak Memory Models* (cf. the relevant section) comprises now three items:

`memevent` (Ocaml, 25000 locs [lines of code]) The model checker. It has been rewritten for flexibility and now integrates about 10 different models for three architectures (X86, Power and ARM).

`litmus` (Ocaml, 4800 locs + C, 500 locs) It takes the same input as `memevents` and produces executables that exercices the memory model of X86 and Power machines. This year, we gained access to 2 supercomputers, which lead us to add a cross compilation feature to `litmus`, and to consider running several instances of the same test concurrently. The throughput of test outcomes have been multiplied by a factor of about 100, allowing us to consider running 1000 tests 10^9 times in a reasonable time.

`diy` (Ocaml, 3500 locs) It is s a new tool, developed by J. Alglave and L. Maranget. `diy` produces inputs for `litmus` and `memevents` from concise specification.

The combination of access to supercomputers hosted at IDRIS (CNRS) and HPCX center (UK), improvements of `litmus` and the design of `diy` allows us to tackle in-deep testing of the Power architecture machines.

L. Maranget is now the main developer of this tools suite.

6. New Results

6.1. Weak Memory Models

Participants: Jade Alglave, Luc Maranget, Giulio Manzonetto, Peter Sewell [U. of Cambridge], Susmit Sekar [U. of Cambridge], 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 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 or Coq.

- J. Alglave and L. Maranget concentrated on the weak memory model of the Power PC architecture. They designed a generic model, which conservatively extends the models of the Sparc hierarchy (TSO/PSO/RMO). Moreover, the expression of the validity condition in these new models is much simpler and more prone to efficient implementation than the original one. Additionnaly, in spite of its conceptual simplicity and of being global-time based, our experiments demonstrate that this generic model yields valid insights on the Power architecture.

  Our experimentats on Power5 lead to some contact with IBM. We pursue an informal collaboration with IBM, based upon weekly telephone conferences with a chief IBM engineer.

  This work is now under publication.

- G. Manzonetto, P. Sewell and F. Zappa Nardelli also began the investigation of equivalences of programs running on top of relaxed memories, but this line of research is still at an early stage.

  A position paper[17] has been presented in the workshop EC^2.

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

This work is also done in collaboration with Scott Owens, Tom Ridge, Mark Batty (Cambridge University), and Samin Isthiaq (Microsoft Research Cambridge), Jaroslav Ševčík (Edinburgh University) and Derek Williams (IBM).

6.2. Integrating Typed and Untyped Code in a Scripting Language

Participants: Jan Vitek [Purdue University], Tobias Wrigstad [Purdue University], Francesco Zappa Nardelli.

Many large software systems originate from untyped scripting lan- guage code. While good for initial development, the lack of static type annotations can impact code-quality and performance in the long run.

In collaboration with Jan Vitek and Tobias Wrigstad (Purdue University), we studied an approach for integrating untyped code and typed code in the same system to allow an initial prototype to smoothly evolve into an efficient and robust program. We introduced *like types*, a novel intermediate point between dynamic and static typing. Occurrences of *like types* variables are checked statically within their scope but, as they may be bound to dynamic values, their usage is checked dynamically. Thus *like types* provide some of the benefits of static typing without decreasing the expressiveness of the language.

We provided a formal account of *like types* in a core object calculus and evaluated their applicability in the context of a new scripting language.

A paper describing our approach has been accepted in POPL [16], while the implementation in the Thorn programming language (developed by Purdue University and IBM) is available from http://www.thorn-lang.org.

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 [22], TGC'07 [23] and in a special issue of the Journal of Computer Security [24].

We have added 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 developed 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.

This work was presented in [12].

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 2009 we studied a general scheme to generate audit trails, generalising our past results. 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 defined a generic setup for auditability, and we understood how type-checking can be used to check auditability.

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 [21].

We have used our compiler to verify the Transport Layer Security protocol (TLS). In that work [20], 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. Models and theories of lambda calculus

Participant: Giulio Manzonetto.

The lambda calculus is a paradigmatic programming language that is at the basis of all functional programming languages. Lambda theories (equational extensions of lambda calculus) allow to study all the meaningful notions of program equivalence, and constitute a very rich and complex mathematical structure. In our research we use the denotational models of lambda calculus to study computational properties of programs, and to understand the structure of the set of lambda theories.

In 2009 we have proved that there is an inverse relationship between the effectivity of the models of lambda calculus and the effectivity of the corresponding theories. More precisely, we have shown that no effective model of lambda calculus can have lambda-beta or lambda-beta-eta as equational theory [11]. This can be seen as a partial answer of an open problem posed by Honsell 25 years ago. Moreover, we have defined and studied a new model of lambda calculus, living in a category of sets and (multi-)relations and provided a characterization of its equational theory [14]. We have shown that this model has a rich and powerful structure

that allows to interpret a non-deterministic and parallel extension of lambda calculus [13]. Finally, we have generalized the notion of model of lambda calculus obtaining the class of Church algebras [15], and used them to study the lattices of equational theories. This allows to prove a meta-stone representation theorem whose scope is much bigger than the one of the other representation theorems: it is indeed applicable to all varieties of algebras.

This work was achieved in collaboration with Chantal Berline, Antonio Bucciarelli, Thomas Ehrhard (PPS, Paris), Antonino Salibra (Ca'Foscari, Venice).

6.7. Proofs of Strong Normalisation

Participants: Jean-Jacques Lévy, Giulio Manzonetto.

The usual proofs of strong normalisation are quite enigmatic in the typed lambda calculus, proofs by realisability, reductibility (Tait), *candidats de réductibilité* (Girard), saturated sets (Krivine). We tried to exhibit more intuitive proof techniques. If this can easily be done in first-order typed lambda calculus, it is less clear of System F or other higher-order typed calculi.

In 2009, we started to work actively on this rather fundamental topic, and hope to get results in 2010.

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 2009. In October 2009, he project has been renewed for next 3 years (2010-2012).

8. Dissemination

8.1. Animation of research

J.-J. Lévy is director of the Microsoft Research-INRIA Joint Centre, see http://www.msr-inria.inria.fr. He participated to the renewal of the Microsoft Research-INRIA agreement for next 4 years. He organised the Forum 2009 http://msr-inria.inria.fr/forum2009 on 28 January 2009, whch presented results of the Joint Centre after 2 years of intensive research. In 2009, he chaired 3 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.-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.

J.-J. Lévy co-edited the book in memory of Gilles Kahn [18].

F. Zappa Nardelli is the correspondent of the ANR ParSec Project for the Moscova project-team.

F. Zappa Nardelli is the correspondent of the *equipes associées MM*.

F. Zappa Nardelli served in the CR recruiting committee of INRIA Saclay–Île-de-France.

8.2. Teaching

Our project-team participates to the following courses:

- J. Alglade is instructor (*moniteur*) of the lab courses for courses "Algorithmique Expérimentale" and "Programming in C" at U. of Paris 12 (Créteil), September 2009-January 2010.
- "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);
- "Introduction to the Coq proof assistant", F. Zappa Nardelli January, Purdue University, USA (2 hours).
- "J-O-Caml", Master Course, November-December, Tsinghua University, Beijing (15 students, 12h + final exam) http://jeanjacqueslevy.net/courses/tsinghua/j-o-caml.

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.
- June 16, J.-J. Lévy participated to the workshop about Computer Science for Education and presented a talk *Combien d'objets dans cette image?* to mathematics teachers of the *Académie de Versailles*
- December 9, J.-J. Lévy gave a talk on "Lambda calculus and Programming" in the G. Berry lectures at the Collège de France in Paris.

8.3. Participations to conferences, Seminars, Invitations

8.3.1. Participations to conferences

- January, 3-10, G. Manzonetto attended the LFCS09 conference in Boca Raton (Florida, États-Unis). He gave a talk on "A relational model of a parallel and non-deterministic lambda calculus"
- January 19-23, J. Alglade, P.-M. Deniélou and F. Zappa Nardelli participated to the 36th ACM POPL conference, Savannah, Georgia.
- July 8-10, P.-M. Deniélou, J. Leifer, J.-J. Lévy, J. Planul attended the 22nd IEEE Computer Security Foundations Symposium (CSF), Port Jefferson, New York. P.-M. Deniélou presented the paper "Cryptographic Protocol Synthesis and Verification for Multiparty Sessions".
- July 13-14, J.-J. Lévy participated to the Faculty Summit 2009, Microsoft Conference Center, Redmond, USA.
- August, 23-38, G. Manzonetto attended the MFCS09 conference in Novy Smokovek (Slovakia). He gave a talk on "A general class of models of $\mathcal{H}^{\star}$".
- September 1-4, J. Alglade and J. Planul participated to the 20th CONCUR conference, Bologne. J. Planul presented....

- Sep 21-22, N. Guts presented a paper “Reliable Evidence: Auditability by Typing” at the 14th European Symposium on Research in Computer Security (ESORICS), Saint-Malo.

8.3.2. Other Talks and Participations

- January, 9-30, F. Zappa Nardelli visited Purdue Univerity (États-Unis), for collaboration with Jan Vitek. He also gave lectures on "Introduction to the Coq Proof Assistant".
- February, 15-22, G. Manzonetto attended the kickoff meeting of the project CONCERTO in Bologna. He gave a talk on “From lambda calculus to universal algebra, and back”.
- February, 24-26, J.-J. Lévy participated to TechFest, Microsoft Research, Redmond, USA.
- Mar 9-11, J. Alglade and L. Maranget visited P. Sewell and S. Sekar at the Computer laboratory, Cambridge, UK.
- March, 9-13, G. Manzonetto and F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- May, 3-8, G. Manzonetto visited Barendregt and Klop at Nijmegen and Amsterdam (Holland). He gave a talk on “Models and theories of lambda calculus” at Vrije Universiteit and a talk on “Applying universal algebra to lambda calculus” at Radboud University.
- May 8-August 8, N. Guts was intern at Microsoft Research, Cambridge (Work with K. Bhargavan, A. Gordon and C. Fournet).
- May 25, F. Zappa Nardelli gave a talk on “Integrating Typed and Untyped Code in a Scripting Language” at the Gallium-Moscova seminar of INRIA Paris-Rocquencourt.
- June 3-5, F. Zappa Nardelli attended the fourth meeting of the ANR ParSec Project at IRISA, Rennes. He gave a talk on “Integrating Typed and Untyped Code in a Scripting Language”.
- June, 11-12, J.-J. Lévy participated to a meeting of Microsoft Research Joint Centers at the Space Research Institute (IKI) in Moscow.
- June, 25-29, F. Zappa Nardelli attended the EC2 workshop in Grenoble. He gave a talk on “Memory Models must be Rigourous”.
- August, 4-16, J. Planul attended the Marktoberdorf summer school on “Logics and Languages for Reliability and Security”, Martkoberdorf, Germany.
- August, 13-14, G. Manzonetto gave a talk on “Applying universal algebra to lambda calculus” at the University of Leicester.
- September, 3, G. Manzonetto gave a talk on “A relational model of lambda calculus, and beyond” at the University of Edinburgh.
- October, 4-7, F. Zappa Nardelli visited the Computer Laboratory of the University of Cambridge for collaboration with Peter Sewell.
- October, 6, K. Bhargavan, J. Leifer, J. Planul gave demos of research made within our collaboration with Microsoft Research, in presence of S. Ballmer, CEO MS, amd V. Pécresse, Minister of Research, at the inauguration day of the new MS building in Paris,
- October 12-December 4, J.-J. Lévy visited Tsinghua University in Beijing (group of J.-P. Jouannaud). He attended the TAB (Technical Advisory Board) meeting and the workshop on Verified Software at Microsoft Research Asia. He gave a talk at the Institute of Software of the Academy of Science.
- October 22, F. Zappa Nardelli gave a talk on “Integrating Typed and Untyped Code in a Scripting Language” at PPS, Paris 7.
- November 24, J. Alglade gave a talk at the PPS laboratory, U. of Paris 7, on *Weak Memory Models*.
- December 4, J. Alglade gave a talk at the U. of Singapore, on *Weak Memory Models*, while visiting Aquinas Hobor.

8.3.3. Visits

- November 12-17, Suresh Jagannathan visited Rocquencourt for collaboration with F. Zappa Nardelli. He gave a talk on “Serializability Enforcement for Concurrent ML”.
- November 2, J. Leifer hosted Yuri Gurevich (Microsoft Research) and organised his talk jointly with LIX *The Church-Turing Thesis: Story and Recent Progress*.
- November 29-December 2, Tobias Wrigstad visited Rocquencourt for collaboration with F. Zappa Nardelli.

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", 2003, http://hal.inria.fr/inria-00071732/fr/, Extended Abstract of INRIA Research Report 4851.

[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

Articles in International Peer-Reviewed Journal

[11] C. BERLINE, G. MANZONETTO, A. SALIBRA. *Effective Lambda Models Versus Recursively Enumerable Lambda Theories*, in "Mathematical Structures in Computer Science", vol. 19, n^{o} 5, October 2009, p. 897-942.

International Peer-Reviewed Conference/Proceedings

[12] K. BHARGAVAN, R. CORIN, P.-M. DENIÉLOU, C. FOURNET, J. J. LEIFER. *Cryptographic Protocol Synthesis and Verification for Multiparty Sessions*, in "CSF", IEEE Computer Society, 2009, p. 124-140, http://doi.ieeecomputersociety.org/10.1109/CSF.2009.26.

[13] A. BUCCIARELLI, T. EHRHARD, G. MANZONETTO. *A relational model of a parallel and non-deterministic lambda-calculus*, in "Lecture Notes in Computer Science", vol. 5407, 2009, p. 107-121.

[14] G. MANZONETTO. *A general class of models of* $\mathcal{H}^{\star}$, in "Mathematical Foundations of Computer Science (MFCS'09)", Lecture Notes in Computer Science, vol. 5734, Springer, 2009, p. 574-586.

[15] G. MANZONETTO, A. SALIBRA. *Lattices of Equational Theories as Church Algebras*, in "Proc. 7th Panhellenic Logic Symposium", C. DROSSOS, P. PEPPAS, C. TSINAKIS (editors), Patras University Press, 2009, p. 117-121.

[16] T. WRIGSTAD, F. ZAPPA NARDELLI, S. LEBRESNE, J. OSTLUND, J. VITEK. *Integrating Typed and Untyped Code in a Scripting Language*, in "Proc. POPL 2010", 2010, to appear.

[17] F. ZAPPA NARDELLI, P. SEWELL, J. ŠEVČÍK, S. SARKAR, S. OWENS, L. MARANGET, M. BATTY, J. ALGLAVE. *Relaxed memory models must be rigorous*, in "Proceedings of Exploiting Concurrency Efficiently and Correctly – (EC)2. CAV 2009 Workshop. Grenoble, France", June 2009, 4pp.

Scientific Books (or Scientific Book chapters)

[18] Y. BERTOT, G. HUET, J.-J. LÉVY, G. PLOTKIN. *From Semantics to Computer Science: Essays in Honour of Gilles Kahn*, Cambridge University Press, New York, NY, USA, 2009.

Other Publications

[19] P. SEWELL, F. ZAPPA NARDELLI, S. OWENS, G. PESKINE, T. RIDGE, S. SARKAR, R. STRNIŠA. *Ott: Effective Tool Support for the Working Semanticist*, 2010, To appear in the Journal of Functional Programming.

References in notes

[20] 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, http://doi.acm.org/10.1145/1455770.1455828.

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

[22] 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/.

[23] 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.

[24] 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, http://dx.doi.org/10.3233/JCS-2008-0334.

[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*, 1997, Available electronically.

[29] 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.